ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 9, Issue, 2(B), pp. 23860-23864, February, 2018

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

SOFTWARE DEFECT PREDICTION USING ENHANCED RELEVANCE VECTOR MACHINE

Vanithamani P* and Jaiganesh V

Department of Computer Science, Dr.N.G.P. College of Arts and Science,
Bharathiar University, Coimbatore, Tamil Nadu, India

DOI: http://dx.doi.org/10.24327/ijrsr.2018.0902.1550

ARTICLE INFO

Article History:

Received 20th November, 2017
Received in revised form 29th December, 2017
Accepted 30th January, 2018
Published online 28th February, 2018

Key Words:

Data mining, machine learning, support vector machine, relevance vector machine, software defect prediction, Promise Software engineering repository.

ABSTRACT

Data mining using machine learning techniques grabbed the interest of researchers in recent years. Software defect prediction is one of the thrust research area and data mining techniques are applied to identify the defects that are present in the datasets. In this research work an enhanced relevance vector machine (ERVM) is used for software defect prediction. From the extensive literature study it is observed that relevance vector machine classifier is comparably delivering better performance than that of support vector machine. This implication is taken for the research work. Datasets are collected from Promise software engineering repository [25] that has a collection of publicly available datasets and tools to serve researchers in building predictive software models (PSMs) and software engineering community at large. Two state of the art datasets namely PC1 and CM1 are taken for estimating the efficiency of the RVM and ERVM. MATLAB is used for implementing both RVM and ERVM. Defect prediction accuracy, sensitivity, specificity and time taken for execution are the performance metrics chosen and the results encompasses that ERVM performs better.

INTRODUCTION

Recently people around the world are evidently and impalpably getting benefitted by software industry. Reliance of people on software is propounding too many which results in necessity of quality software. Commonly, the quality measure of the software is perceived by uncovering the defects in it. One of the metric to forecast / predict the software quality is software defect density that tends to analyze the same. The defect density of the software is computed by splitting the dimension of the software with total number of defects in it. In several literatures it is stated that a software defect is a product anomaly. The software reliability of software is described as the probability that software will not be the reason for the crash of a system in for a particular amount of time beneath the precise conditions. Software defect prediction is the primary component during the development of reliable software. In each step of software development life cycle for attaining target defect guesstimate, it is required to predict the defect density indicator. For the past decades several models have been proposed for software reliability estimation and prediction. It is noted that conventional software reliability prediction models are not more triumphant in predicting the reliability of the software. Also such proposed models are not user friendly and many of such models are probabilistic based approaches. To add up with the above said things, failure data are also not available in the early phases of the software development life cycle. Whilst some of them are presented in the form of expert knowledge through certain software metrics. This paper is organized as follows: this section introduces about software defect prediction.

Section 2 briefly presents the related works. Section 3 portrays the background of support vector machine. Section 4 proposes enhanced relevance vector machine classifier for software defect prediction. Section 5 projects the simulation results. Section 6 emphasizes the concluding remarks of the paper.

Related Works

Building defect-related feature space is one among the primary works in software defect classification. In order to classify the software defect-related attributes, several approaches were presented that depends on diverse information [1], namely code metrics [2], process metrics [3], previous defects [4] and more

Corresponding author:* **Vanithamani P
Department of Computer Science, Dr.N.G.P. College of Arts and Science, Bharathiar University, Coimbatore, Tamil Nadu, India

[5,6]. Many literatures mentioned that LOC data performs better in software defect prediction [7, 8]. D'Ambros *et al*. [9] mentioned in their article that change coupling correlates with defects. In [10], the authors Nagappan and Ball [10] applied code churn and also dependency metrics in order to predict defect-prone modules. Khoshgoftaar and Seliya [11] had chosen 14 process metrics. Weyuker *et al*. [12] built a defect-count prediction model by taking into account of certain number of process measures along with structural measures. Feature selection on group of metrics also implemented by various researchers for improving the performance of the classifier and can be found in [13, 14, 15, 16]. It is to be noted that there exists several statistical models and machine learning algorithms have been applied. In the literature that can be found in [17], Olague *et al*. applied logistic regression to predict defects in software. Mizuno and Kikuno [18] stated in the literature a type of Markov model named Orthogonal Sparse Bigrams suited best for software defect prediction. In the literature [19] Bibi *et al*. implemented Regression through Classification mechansim in software defect prediction. Khoshgoftaar *et al*. [20] constructed a classification tree that is based on TREEDISC algorithm for predicting module that was likely to have defects which is based on software product, process, and execution metrics. Early software defects prediction using fuzzy logic is proposed in [21]. Software defect prediction using Bayesian networks can be found in the literature [22]. Combining the requirement information for software defect estimation in design time has been implemented in the paper [23]. Incremental updating approximations in probabilistic rough sets under the variation of attributes are discussed in [24].

Enhanced Relevance Vector Machine for Software Defect Prediction

As one of the state-of-the-art tools for machine learning, support vector machines (SVMs) produce a model function dependent only on a subset of kernel basis functions associated with some of the training samples, i.e., support vectors. Although the SVM produces a sparse model, in some cases the number of support vectors is still quite large. Relevance vector machine (RVM) is a new supervised learning method proposed in 2001 by Tipping[3], which can be used to solve regression and classification problems, resulting from their sparsity and mathematical tractability. Compared with SVM, RVM can have probabilistic output and more sparse, and kernel function to choose more freely and so on. The Bayesian formulation of the RVM provides a better sparsity, as well as an automatic estimation of hyper parameters and probabilistic outputs, while predicting accuracy is not traded off.

The RVM can be categorized into a pruning method of the RBF networks, using kernel function as the candidate basis functions, i.e., $\phi_i(x) = k(x, x_i)$, which defines one basis function for each example in the training set $X = \{x_i\}^k{}_{i=1}$. The majority of the weights are trained to be zero, while only those "relevant vectors" corresponding non-zero weights are selected to construct the model in Equation (1). To achieve such a sparsity but avoid overfitting, the RVM imposes a spherical Gaussian prior on the weights, and a Gamma hyperprior distribution for the variance. Given an input x, the output of the RVM is that of the form

$$y(x:w) = \sum_{i=1}^{k} w_i \phi_i(x) + w_0 \quad \ldots (1)$$

The performance of the original RVM in terms of sparsity is determined by the smoothness of the prior. However the lack of an explicit prior structure over the weight variance means that the sparsity actually depends on the choice of kernel functions and/or kernel parameters. This may lead to severe overfitting or underfitting. To control sparsity in Bayesian regression by incorporating a flexible noise dependent smoothness prior to replace the Gamma prior in the classic RVM, the symmlets with smoothness prior make the RVM regression suitable for a large variety of signals, requiring no additional kernel parameters to be determined by cross-validation. The objective of the E-RVM learning is to obtain the best set of the relevance vectors in terms of goodness of fit to data, as well as the sparsity, i.e. the least number of relevance vectors.

Considering the three random processes, x, t, j, in the RVM training phase, the joint distribution among them can be calculated by either of these two formulae:

$$p(x,t,j) = p(x)p(t \mid x)p(j \mid t,x) \quad \ldots (2)$$

$$p(x,t,j) + p(j)p(x \mid j)p(t \mid x,j) \quad \ldots (3)$$

In the ideal scenarios, Equation (2) and Equation (3) would return the same result, however, they are not so unless j corresponds to the optimal set of the relevance vectors. The breakdown of the right hand sides of Equation (2) indicates that it is a forward pathway to obtain the solution from the known training data and targets, whereas Equation (3) follows the backward pathway to test the performance of current solution.

When we use $p_{train}(x,t,j)$ and $p_{test}(x,t,j)$ to denote the two joint probabilities of Equations (2) and (3) respectively, the best match between them, corresponding to the popular mean square error (MSE) and maximum likelihood(ML), can be achieved by the minimization of the classic Kullback-Leibler divergence:

$$KL(p_{train} \mid p_{test}) + \int_{x,t,j} p_{train}(x,t,j) \log \frac{p_{train}(x,t,j)}{p_{test}(x,t,j)} d_x d_t d_j \quad \ldots (4)$$

Which can push $ptrain(x,t,j)$ to match $p_{test}(x,t,j)$, such that the solution will fit the training data most. However, the problem with the KL-based learning is the low generalization ability due to the lack of control on least complexity. In Equation (4), the distribution of p_{train} is the fixed prior reference distribution, and the distribution of p_{test} is optimized to match p_{train} under the KL divergence minimization. The following negative cross entropy will be maximized to construct the RVM:

$$\mathbf{H}(p_{train} \| p_{test}) = \int_{x,t,j} p_{train}(x,t,j) \log p_{test}(x,t,j) d_x d_t d_j \quad \ldots (5)$$

Comparing Equations (4) and (5), we can see that $H(p_{train}\| p_{test}) = -KL(p_{train}\| p_{test}) + H(p_{train}\| p_{train})$ which indicates that the maximization of Train-Testentropy consists of two parts, namely, the minimization of KL divergence between Test and Training, and the maximization of entropy in Training space. The former leads to the goodness of fit to data, whereas the latter leads to the least complexity under some constraints. As the training data are in the paired form (x_i, t_i), particularly the model $ptrain\ (x, t, j) = p(x, t)p(j \mid t, x)$ is considered with the following structures

$$p(x,t) = \frac{1}{N}\sum_{n=1}^{N}\delta(x - x_n)\delta(t - t_n) \quad \ldots (6)$$

$p(j \mid t, x)$ is free $\quad \ldots (7)$

In the backward testing phrase, the "true" probability distribution is $ptest(x,t,j) = p(x)p(x \mid j)p(t \mid x, j)$ corresponding to the backward testing pathway, where the ERVM constructed by the candidate relevance vectors j is tested whether it outputs the target t given an input x. It is denoted as

$$p(j) = p_j \quad \ldots (8)$$

which is subject to $pj \geq 0$ and $\sum p_j = 1$. 1 denotes the software defect and 0 denotes software non-defect.

RESULTS

PC-1 dataset contains 1109 records. Out of that 1109 records 1032 records doesn't contain software defects. Remaining 77 records contain software defects. CM-1 dataset contains 498 records. Out of those 498 records 450 records doesn't contain software defects. Remaining 48 records contain software defects. It is shown in Table 1. Simulation results for PC1 dataset is portrayed in Table 2. Simulation results for CM1 dataset is depicted in Table 3. In Figure 1 the prediction accuracy is portrayed. From the simulation results it is observed that ERVM classifier performs better in terms of predicting software defects.

Table 1 Dataset Information

Dataset	Total Records	Defects	Non-Defects
PC-1	1109	77	1032
CM-1	498	48	450

Table 4 Time Taken for Execution of RVM and ERVM for the Datasets

	RVM	ERVM
PC1 Dataset	59.06 seconds	41.33 seconds
CM1 Dataset	27.34 seconds	19.04 seconds

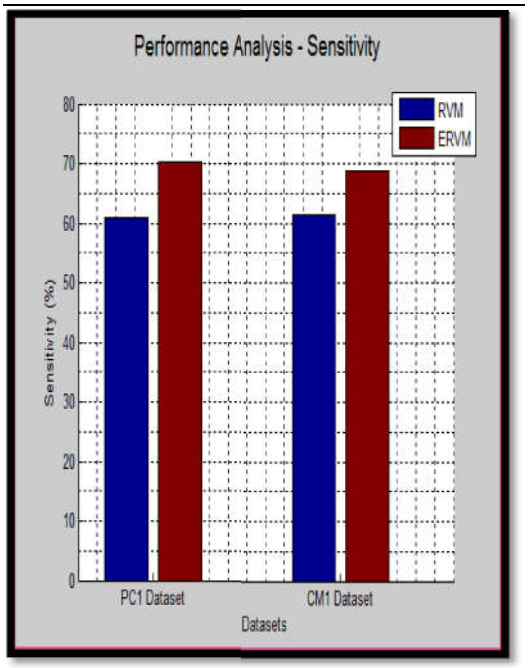

Fig 1 Performance Analysis - Sensitivity

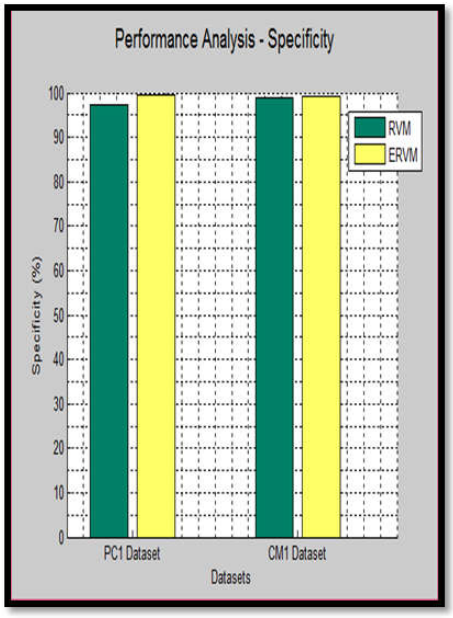

Fig 2 Performance Analysis – Specificity

Table 2 Simulation Results for PC1 Dataset

Algorithm	True Positive (Correctly Classified as Defects)	False Positive (Incorrectly Classified as Defects)	True Negative (Correctly Classified as Non-Defects)	False Negative (Incorrectly Classified as Non-Defects)	Sensitivity	Specificity	Accuracy
RVM	50	27	1000	32	60.97	97.37	94.67
ERVM	73	4	1001	31	70.19	99.60	96.84

Table 3 Simulation Results for CM1 Dataset

Algorithm	True Positive (Correctly Classified as Defects)	False Positive (Incorrectly Classified as Defects)	True Negative (Correctly Classified as Non-Defects)	False Negative (Incorrectly Classified as Non-Defects)	Sensitivity	Specificity	Accuracy
RVM	43	5	423	27	61.42	98.83	93.57
ERVM	44	3	431	20	68.75	99.30	95.38

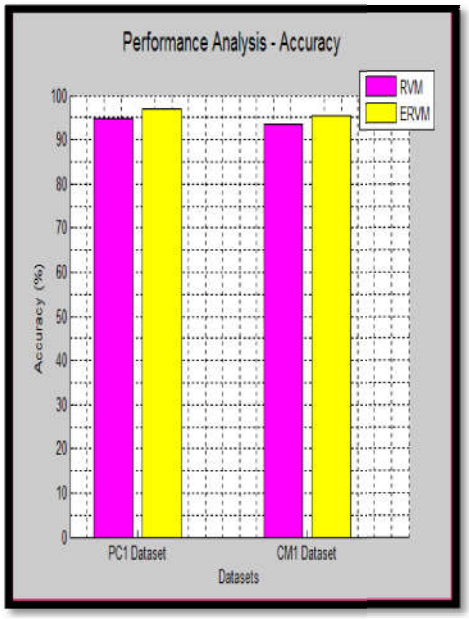

Fig 3 Performance Analysis - Accuracy

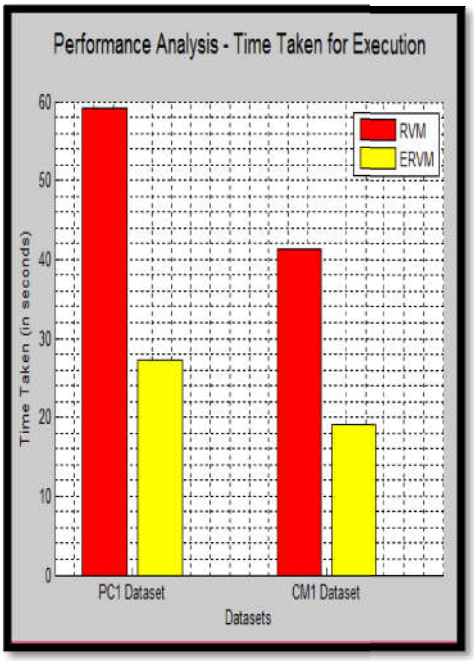

Fig 4 Performance Analysis - Time Taken for Execution

CONCLUSIONS

This research work aims to propose an enhanced relevance vector machine classifier for software defect prediction. Two datasets are obtained from Promise software engineering repository. Kullback-Leibler divergence factor is included in ERVM in order to reduce the time complexity of the classifier. Implementations are carried out using MATLAB. The obtained results proved that the proposed ERVM consumes less time than that of RVM and also accuracy has been improved.

References

1. M. D'Ambros, M. Lanza, R. Robbes, Evaluating defect prediction approaches: a benchmark and an extensive comparison, Empir. Softw. Eng. 17 (4) (2012) 531-577.
2. N. Nagappan, T. Ball, A. Zeller, Mining metrics to predict component failures, in: Proceedings of the 2006 International Conference on Software Engineering, 2006, pp. 452-461.
3. A.E. Hassan, Predicting faults using the complexity of code changes, in: Proceedings of the 2009 International Conference on Software Engineering, 2009, pp. 78-88.
4. S. Kim, T. Zimmermann, E. Whitehead Jr, A. Zeller, Predicting faults from cached history, in: Proceedings of the 2007 International Conference on Software Engineering, 2007, pp. 489-498.
5. E. Arisholm, L.C. Briand, M.J. Fuglerud, Data mining techniques for building faultproneness models in telecom java software, in: Proceedings of the 18th IEEE International Symposium on Software Reliability Engineering, 2007, pp. 215-224.
6. T.J. Ostrand, E.J. Weyuker, How to measure success of fault prediction models, in: Proceedings of the Fourth InternationalWorkshop on Software Quality Assurance: in Conjunction with the 6th ESEC/FSE Joint Meeting, 2007, pp. 25-30.
7. H.Y. Zhang, An investigation of the relationships between lines of code and defects, in: Proceedings of 2009 IEEE International Conference on Software Maintainance, 2009, pp. 274-283.
8. Y.M. Zhou, B.W. Xu, H. Leung, On the ability of complexity metrics to predict faultprone classes in object-oriented systems, *J. Syst. Software* 83 (4) (2010) 660-674.
9. M. D'Ambros, M. Lanza, R. Robbes, On the relationship between change coupling and software defects, in: Proceedings of the 16thWorking Conference on Reverse Engineering, 2009, pp. 135-144.
10. N. Nagappan, T. Ball, Using software dependencies and churn metrics to predict field failures: An empirical case study, in: Proceedings of First International Symposium on Empirical Software Engineering and Measurement, 2007, pp. 364-373.
11. T.M. Khoshgoftaar, N. Seliya, Analogy-based practical classification rules for software quality estimation, Empir. Softw. Eng. 8 (4) (2003) 325-350.
12. E.J. Weyuker, T.J. Ostrand, R.M. Bell, Using developer information as a factor for fault prediction, in: Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering, 2007, pp. 1-7.
13. C. Bird, N. Nagappan, H. Gall, B. Murphy, P. Devanbu, Putting it all together: using socio-technical networks to predict failures, in: Proceedings of the 20th International Symposium on Software Reliability Engineering, 2009, pp. 109-119.
14. T.M. Khoshgoftaar, K. Gao, N. Seliya, Attribute selection and imbalanced data: problems in software defect prediction, in: Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence, 1, 2010, pp. 137-144.
15. M.X. Liu, L.S.Miao, D.Q. Zhang, Two-stage cost-sensitive learning for software defect prediction, IEEE T. Reliab. 63 (2) (2014) 676-686.

16. S. Shivaji, E.J. Whitehead Jr, R. Akella, S. Kim, Reducing features to improve code change-based bug prediction, IEEE T. Softw. Eng. 39 (4) (2013) 552-569.
17. H.M. Olague, L.H. Etzkorn, S. Gholston, S. Quattlebaum, Empirical validation of three software metrics suites to predict fault-roneness of object-oriented classes developed using highly iterative or agile software development processes, IEEE T. Softw. Eng. 33 (6) (2007) 402-419.
18. O. Mizuno, T. Kikuno, Training on errors experiment to detect fault-prone software modules by spam filter, in: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2007, pp. 405-414.
19. S. Bibi, G. Tsoumakas, I. Stamelos, I.P. Vlahavas, Software defect prediction using regression via classification, in: Proceedings of IEEE International Conference on Computer Systems and Applications, 2006, pp. 330-336.
20. T.M. Khoshgoftaar, X.J. Yuan, E.B. Allen, W.D. Jones, J.P. Hudepohl, Uncertain classification of fault-prone software modules, Empir. Softw. Eng. 7 (4) (2002) 297-318.
21. D.K. Yadav, S.K. Charurvedi, R.B. Mishra, Early software defects prediction using fuzzy logic, *Int. J. Performability Eng*. 8 (4) (2012) 399-408.
22. Okutan, O.T. Yildiz, Software defect prediction using Bayesian networks, Empirical Softw. Eng. 19 (1) (2014) 154-181.
23. Y. Maa, S. Zhua, K. Qin, G. Luo, Combining the requirement information for software defect estimation in design time, Inform. Process. Lett. 114 (9) (2014) 469-474.
24. D. Liu, T.R. Li, J.B. Zhang, Incremental updating approximations in probabilistic rough sets under the variation of attributes, Knowl-Based Syst. 73 (2015) 81-96.
25. PC1 and CM1 dataset - http://promise.site.uottawa.ca/ SERepository

How to cite this article:

Vanithamani P and Jaiganesh V.2018, Software Defect Prediction Using Enhanced Relevance Vector Machine. *Int J Recent Sci Res.* 9(2), pp. 23860-23864. DOI: http://dx.doi.org/10.24327/ijrsr.2018.0902.1550

****