Journal of Communications Technology, Electronics and Computer Science, Issue 5, 2016
ISSN 2457-905X

Improvement in Bug Localization based on Kernel Extreme Learning Machine

Marzieh Rahmati, Mohammad Ali Zare Chahooki
Electrical and Computer Engineering Department
Yazd University
Yazd, Iran
rahmati@stu.yazd.ac.ir, chahooki@yazd.ac.ir

***Abstract*— Bug localization uses bug reports received from users, developers and testers locate buggy files. Since finding a buggy file among thousands of files is time consuming and tedious for developers, various methods based on information retrieval is suggested to automate this process. In addition to information retrieval methods for error localization, machine learning methods are used to. Machine learning-based approach, improves methods of describing bug report and program code by representing them in feature vectors. Learning hypothesis on an Extreme Learning Machine (ELM) has been recently effective in many areas. This paper shows the effectiveness of non-linear kernel of ELM in bug localization. Furthermore the effectiveness of Different kernels in ELM compared to other kernel-based learning methods are analyzed. The experimental results for hypothesis evaluation on Mozilla Firefox dataset show the effectiveness of Kernel ELM for bug localization in software projects.**

Keywords*— machine learning; Extreme learning machine; information retrieval.

I. INTRODUCTION

Quality of software is vital for the success of software project. However a lot of effort is made for increasing the quality of software project, but still maintainers find themselves face with a stream of bug reports. Once a bug report entered the system and approved, developers should investigate buggy source code files and solve them. Using bug report to find buggy files in a large software system with lots of bug reports is time consuming and tedious [1].

There are two methods for bug localization. Some methods are based on test cases and another group is based on bug reports. There are a variety methods based on test cases that do the same thing, but in a different manner [2], [3], [4], [5]. The second group that based on bug reports are worked in two categories, information retrieval and machine learning. In recent years, researchers have been using information retrieval methods to find buggy files. They suppose the bug reports as a query and rank the source code files according to the relationship between the bug report and source code files. Developers examine the returned files and solve the defect [6], [7]. In machine learning based method, bug report represents as a feature vector and source code files is the class labels of dataset [1].

In this paper, we use the content of bug report and different methods based on machine learning to predict buggy files. Although machine learning based method used in a variety researches area, but there are few researches on bug localization based on bug reports that uses machine learning methods [1]. Because of this, we focus on machine learning methods. Therefore, first, we extract different information such as summary, description and meta-data from bug reports. We train our model with 70 percent of bug reports then test it with the remaining data. Since Extreme Learning Machine (ELM) and the kernel-base of ELM have been used in different applications of machine learning and in most of the applications have better result than other kernel-based method such as Support Vector Machine (SVM), in this paper we use Kernel Extreme Learning Machine (KELM) [8] to locate the bug. Since ELM method like SVM method are developed in different kernels, in this paper KELM is compared with kernel-based SVM methods. The evaluation result of this research on Mozilla dataset showed the effectiveness of RBF kernel in ELM method.

The remainder paper is organized as follows: section 2 describes ELM method, section 3 describes Evaluation results and section 4 discusses conclusion and future work.

II. EXTREME LEARNING MACHINE

In this section we summarize the method of ELM [8]. In two recent decades, SVM has been used in many applications of machine learning. It is important that ELM works on single hidden layer feedforward network (SLFN). In ELM, hidden layer doesn't need to be tune and transfer function of this layer is specific.

A. ELM models based on constraint-optimization

First time ELM developed for single hidden layer feedforward neural networks and after that extended to "generalized" SFLN that neurons doesn't look like each other. Fig. 1 shows SLFNs that has a L hidden node.

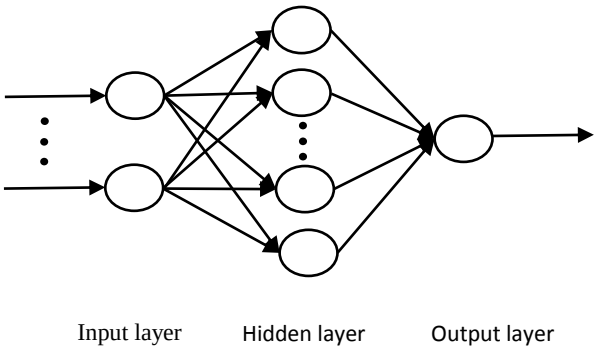

Fig. 1. Single hidden layer of the neural network.

The output function of SLFN will be like (1):

$$f(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta \quad (1)$$

In this equation, function $h(x)$ mapped to d-dimension input space to L-dimension hidden layer feature space. β is a weighted vector between hidden layer and output layer.
For example, if we had binary class classification, decision function of ELM will be like (2):

$$f(x) = sign(h(x)\beta) \quad (2)$$

In contrast to the common learning method, is that ELM not only tries to minimize the training error, but also try to minimize the norm of output weight. Based on Bartlett's theory [9], minimizing the output norm in addition to minimizing train error bring better generalization in performance. ELM function that tries to minimize training error and norm of weight will be like (3):

$$Minimize: ||H\beta - T||^2 \; and \; ||\beta|| \quad (3)$$

In (3) H is hidden layer matrix and define as follow:

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \dots & h_L(x_1) \\ \vdots & \vdots & \vdots \\ h_1(x_N) & \vdots & h_L(x_N) \end{bmatrix} \quad (4)$$

In fact minimizing $||\beta||$ is equivalent to maximizing the margin between two class in binary class classification that means maximizing the $\frac{2}{||\beta||}$.
The minimal norm least square method has been used in ELM method for optimization as follow:

$$\beta = H^{\dagger}T \quad (5)$$

In (5) $H^{\dagger}$ is the Moore–Penrose generalized inverse of matrix H [35], [36]. The orthogonal projection method is one of ways that calculate Moore–Penrose generalized inverse of matrix H.

This method is used when $H^T H$ isn't a singular matrix and we have $H^{\dagger} = (H^T H)^{-1} H^T$ OR HH^T that is not singular and $H^{\dagger} = H^T (HH^T)^{-1}$ is true. According to the ridge regression theory, one can add a positive value to the diagonal of $H^T H$ or HH^T; the result tends to have better generalization performance. So the output weight can be like (6) and (7):

$$\beta = H^T \left(\frac{1}{c} + HH^T\right)^{-1} T \quad ,$$
$$f(x) = h(x)\beta = h(x)H^T \left(\frac{1}{c} + HH^T\right)^{-1} T \quad (6)$$

$$\beta = \left(\frac{1}{c} + H^T H\right)^{-1} H^T T \quad ,$$
$$f(x) = h(x)\beta = h(x)\left(\frac{1}{c} + H^T H\right)^{-1} H^T T \quad (7)$$

In contrast to SVM methods in ELM mapping function has been known [10]. Various form of mapping function in SLFNs is shown in Table I.

TABLE I. DIFFERENT MAPPING FUNCTION FOR SLFNS

Name of function	Equations
Sigmoid function	$G(a,b,x) = \frac{1}{1+\exp(-(a.x+b))}$
hard_limit function	$G(a,b,x) = \begin{cases} 1, & if\ a.x - b \geq 0 \\ 0, & otherwise. \end{cases}$
Gaussian function	$G(a,b,x) = \exp(-b\|x-a\|^2)$
multiquadric function	$G(a,b,x) = (\|x-a\|^2 + b^2)^{\frac{1}{2}}$

B. Different form of ELM

ELM has different models for working with different form of data. Remainder of this section present the equations about classification with one output and multi output. After that using kernels for ELM improvement will be discussed.

1) ELM multiclass classification with single output

ELM can estimate every continues function. Because of this feature, one possible way for multiclass classification is to label the output with attention to ELM output. Equation (8) show the optimization problem.

$$Minimize\ L_{Primal-ELM} = \frac{1}{2}\left|\left|\beta\right|\right|^2 + C\frac{1}{2}\sum_{i=1}^{N} {\xi_i}^2$$
$$subject\ to: h(x_i)\beta = t_i - \xi_i \quad i = 1, \dots, N \quad (8)$$

For solving ELM, we should solve the dual optimization problem that is shown in (9).

$$Minimize\ L_{Dual,ELM} = \frac{1}{2}||\beta||^2 + C\frac{1}{2}\sum_{i=1}^{N}{\xi_i}^2 - \sum_{i=1}^{N}\alpha_i(h(x_i)\beta - t_i + \xi_i) \quad (9)$$

2) multi class classification with multi output

In this case, we have *m* class and *m* output. If one class was p which we see as follow:

$$t_i = [0, \dots, \overset{p}{\hat{1}}, \dots, 0]^T \quad (10)$$

In this case optimization problem is like an equation (11);

$$Minimize\ L_{Primal-ELM} = \frac{1}{2}\left|\left|\beta\right|\right|^2 + C\frac{1}{2}\sum_{i=1}^{N}{\xi_i}^2 \quad (11)$$
$$subject\ to: h(x_i)\beta = {t_i}^T - {\xi_i}^T \quad i = 1, \dots, N$$

Its dual of an optimization problem is like (12):

$$Minimize\ L_{Dual,ELM} = \frac{1}{2}||\beta||^2 + C\frac{1}{2}\sum_{i=1}^{N}{\xi_i}^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}\left(h(x_i)\beta_j - t_{i,j} + \xi_{i,j}\right) \quad (12)$$

3) ELM based on kernel

If mapping function was unknown, we can use kernel in ELM. Equation (13) show the kernel base-ELM.

$$\Omega_{ELM} = HH^T : \Omega_{ELM\ i,j} = h(x_i).h(x_j) = K(x_i, x_j) \quad (13)$$

In this way output function can be like (14):

$$f(x) = h(x)H^T\left(\frac{I}{C} + HH^T\right)^{-1}T = \begin{bmatrix}K(x,x_1)\\ :\\ K(x,x_N)\end{bmatrix}^T\left(\frac{I}{C} + \Omega_{ELM}\right)^{-1}T \quad (14)$$

As you know, in SVM binary class classification training data split with line. This method works when the problem was linear but if the problem was nonlinear, instead of using nonlinear model, transfer the problem to new space and using linear model to solve the problem. Actually the linear model in new space is equivalent to nonlinear model in original space. This paper cover five kernel function in Table II.

TABLE II. DIFFERENT KERNELS USE IN THIS PAPER

Kernel name	***formula***	**Parameter**
Linear	$k(x,y) = x^T y$	NO
RBF (Gaussian)	$k(x,y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$	σ
Polynomial	$k(x,y) = (x^T y + c)^d$	c, d
Sigmoid (Hyperbolic Tangent)	$k(x,y) = \tanh(\propto x^T y + c)$	$\propto$, c
Wavelet	$k(x,y) = \prod_{i=1}^{N} h\left(\frac{x_i - c}{a}\right) h\left(\frac{y_i - c}{a}\right),$ $h(x) = \cos(bx)\exp(-\frac{x^2}{2})$	a, b, c

III. EXPERIMENTAL RESULTS

In this section, ELM with five kernels which discuss in Table IV compare with each other. After that for seeing the effectiveness of ELM, this method compares to kernel base SVM and feed forward neural network.

The Dataset that is used for experimental result is one of the Firefox modules that contain bug reports [1].

A. Dataset

As mentioned before, dataset that used in this paper is bug reports of the Bookmark module in Firefox. The reason of using this dataset is that files have been fixed for each bug report and have been attached to them. Some projects have been needed to have a link between bug report and source code file. This link is for identifying the fixed file, but most of the time there are lots of missing link that causes noisy data [1].

Bug reports in Mozilla are directly reviewed by Mozilla developers. This developer attaches a patch file in bug reports. In the next step these files reviewed by "Core" developers and in final step accepted files submit to bug reports. So these data don't have the problem of missing link and noisy data [1].

Some preprocess need to be done on bug reports. In these bug reports desired feature is extracted from Summary, description and metadata. Metadata contains basic information like bug priority, bug severity, product, component, and etc. these features don't need any preprocess but the text in description and summary need preprocess. For this purpose tokens is extracted from a summary and description, after that they stem and stop words is removed. Feature vector will be frequenting of these extracted tokens. Patch file that is attached to the bug report is extracted for the class label.

The patch is a text file that shows the difference between the original version and fixed version. In this research, Patch file is identified and extracted with designing and implementing a parser [11]. Table V shows the characteristics of data that is used in this research. As you see in the Table VI, there is 1927 samples that first 70 percent base on bug id

is training sample and 30 percent is test sample. There is 557 classes and 6468 features.

TABLE III. CHARACTERISTICS OF DATA

Feature number	Number of classes	Total number of bug reports	period	Module name
6468	557	1927	21-07-2001 – 2-04-2010	**Bookmark**

B. Evaluation result

This section describe the performance measure that we used in our evaluation. Likelihood shows the accuracy of our method. Equation (15) is likelihood.

$$Liklihood = \frac{N_C}{N_C + N_{IC}} \quad (15)$$

In this equation N_C is bug reports that predict correctly and N_{IC} is bug reports that predict incorrectly. Precision shows the number of the files that predict correctly over the number of files that is recommended by our method. Recall shows the number of file that predicted correctly over the number of fixed file. Equation (16) and (17) shows precision and recall.

$$precision = \frac{|F_B \cap F_R|}{|F_R|} \quad (16)$$

$$Recall = \frac{|F_B \cap F_R|}{|F_B|} \quad (17)$$

We donate the set of fixed file to F_B and the number of recommended file to F_R.

C. Result

ELM and SVM with different kernel is examined on desired data. Because ELM is simplification of SVM that execute on single hidden layer feedforward network, this experiment is done on SVM, ELM and SLFN. As we mentioned before we use the first 70 percent of data according to bug id for training set and the remaining for test set. Experiment result for SVM is shown in Table VIII. As you see in Table IX linear kernel have the best result in SVM method.

After that we see the result of feedforward neural network in Table IV. This neural network has two layer that the number of first layer is the number of features and the number of second layer is one. We use different permutation of logsig (logistic sigmoid), tansig(tangent sigmoid), purelin(linear) as transfer function. As we see in Table V result of neural network is very low and the best result is for sigmoid transfer function for first layer and linier transfer function for second layer.

TABLE VI. RESULT OF SVM IN DIFFERENT KERNELS

Likelihood	precision	Recall	kernel
6.3148	4.9094	9.04396	**POLYNOMIAL**
6.3148	4.9094	9.04396	**RBF**
3,7854	3,5861	7.5924	**WAVELET**
16.955	**11,4137**	**18,4229**	**LINEAR**
3,5467	3,3112	6,6154	**SIGMOID**

TABLE VII. RESULT OF SLFN

likelihood	precision	Recall	
0.2595	0.3906	0.7885	**classifier newff+logsig*+logsig****
0.1730	0.1971	0.6001	**classifier newff+tansig+tansig**
0.8996	0.2408	0.5443	**classifier newff+tansig+logsig**
1,4532	0.3640	0.0397	**classifier newff+logsig+tansig**
2,4221	**0.7830**	**2,2540**	**classifier newff+logsig+purelin**
1,4013	0.4057	0.9630	**classifier newff+tansig+purelin**

*transfer function in first layer ;**transfer function in second layer

In this step KELM method is done. Table XIV and XV is shown the result. Table XVI show the best parameter for each kernels in ELM. According to best parameters, Table XVII shows evaluation result for five kernels. As we see in the table, ELM method have better result compared to SVM. As you see in the table RBF kernel has the best result in ELM. In Fig. 2 ELM and SVM compared to each other. As you see RBF kernel of ELM has the best result.

TABLE VIII. DIFFERENT PARAMETERS OF ELM

Parameter3	Parameter2	Parameter1	λ	
-	0.2	1	248	**POLYNOMIAL**
-	-	19	0.9	**RBF**
1	5	4	100	**WAVELET**
-	-	1	1	**LINEAR**
-	10	5	300	**SIGMOID**

TABLE IX. RESULT OF DIFFERENT KERNEL IN ELM

Likelihood	precision	Recall	
32.093	16.581	31.514	**POLYNOMIAL**
32.266	**16.60**	**31.55**	**RBF**
20.7536	10.3022	18.3391	**WAVELET**
25.8823	13.0911	24.9476	**LINEAR**
16.4879	10.0139	20.3978	**SIGMOID**

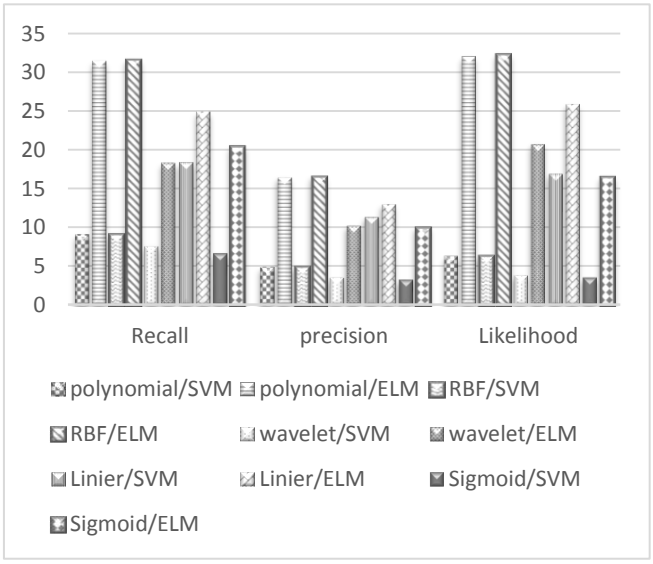

Fig 2. Comparison between SVM and ELM.

IV. Conclusion

Trying to keep the software quality is vital in software projects. Despite all these effort, developers face with lots of bug reports. Since fault localization is time consuming and overwhelming, researchers trying for automating the bug localization process. In this paper we use kernel base extreme learning machine for bug localization.

In future research, we can use the various method of extreme learning machine to improve the results. Also, we can use multiple kernel extreme learning machine for improving the result.

References

[1] D. Kim, Y. Tao, S. Kim, and A. Zeller, "Where should we fix this bug? A two-phase recommendation model," IEEE transactions on software Engineering, vol. 39, no. 11, pp. 1597-1610, 2013.

[2] M. Weiser, "Program Slicing," Proc. Of the 5th international conference on software engineering, no. 4, 1984, pp. 352-357

[3] H. ledgard and M. Weiser, "Programmers Use Slices when Debugging," Communications of ACM, vol. 25, no. 7, pp. 446-452, 1982.

[4] Korel, Bogdan, and Janusz Laski. "STAD-A system for testing and debugging: User perspective." Proc. of the Second workshop on software testing, verification, and analysis, 1988, pp. 13 - 20.

[5] C. Liu, et. al., "Statistical debugging: A hypothesis testing-based approach," IEEE Transactions on software engineering, vol. 32, no. pp. 831-848, 2006.

[6] W. Lafayette, "Retrieval from Software Libraries for Bug Localization : A Comparative Study of Generic and Composite Text Models," In proc. of the 8th working conference on mining Software repositories, 2011, pp. 43-52.

[7] J. Zhou, H. Zhang, and D. Lo, "Where Should the Bugs Be Fixed ? More Accurate Information Retrieval Based Bug Localization Based on Bug Reports," 34th international conference of software engineering (ICSE), 2012, pp. 14-24.

[8] G. Huang, S. Member, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," vol. 42, no. 2, pp. 513–529, 2012.

[9] G. H. Â, Q. Zhu, and C. Siew, "Extreme learning machine : Theory and applications," vol. 70, pp. 489–501, 2006.

[10] N. Cristianini and J. Shawe-Taylor, "An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods." New York, NY, USA: Cambridge University Press, 2000.

[11] N. Bettenburg, et. al., "Extracting Structural information from bug reports," Proceeding of the 2008 international working conference on mining software repositories, ACM, 2008.