

Prediction of Level of Severity of Faults in Software Systems using Density Based Clustering

Parvinder S. Sandhu¹, Sheena Singh², Neha Budhija¹⁺

1 Deptt. Of CSE, Rayat & Bahra Institute of Engg. & Bio-Tech., Mohali, India

2 Deptt. Of CSE, RIEIT, Rail Majra, Punjab, India

1+ Dept. of CSE, Indo Global College of Engineering, Abhipur, India

Abstract. There is lot of work done in prediction of the faults and fault proneness in the various kinds of software systems. But, it is the impact or level of severity of those faults which is more important than number of faults existing in the systems, as the major faults matters most for a developer than the minor ones and these major faults needs immediate attention. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is most widely used density based algorithm and has played a significant role in finding non linear shapes structure based on the density in various application domains. In this paper, the comparative analysis of performance of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Neuro-Fuzzy System is done for prediction of level of severity of faults in the object oriented software systems and NASA's public domain defect dataset is used.

Keywords: Clustering, Decision Tree, K-means, software quality.

1. Introduction

As the majority of faults are found in a few of its modules so there is a need to investigate the modules that are affected severely as compared to other modules and proper maintenance need to be done in time especially for the critical applications [1]. There is lot of work done in prediction of the fault proneness of the software systems. But, it is the severity of the faults that is more important than number of faults existing in the developed system as the major faults matters most for a developer and those major faults needs immediate attention [25].

Prediction of impact of faults in object oriented software systems supports software quality engineering through improved scheduling and project control. It can be a key step towards steering the software testing and improving the effectiveness of the whole process and enables effective discovery and identification of defects. It enables the verification and validation activities focused on critical software components and can be used to improve software process control and achieve high software reliability. Hence, Prediction of impact of faults in object oriented software systems can be used to direct cost-effective quality enhancement efforts to modules.

Cluster analysis or clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields, including machine learning, data mining, pattern recognition, image analysis, information retrieval, and bioinformatics [4]. Hence, clustering can be experimented for the Prediction of impact of faults in object oriented software systems. Clustering algorithms can be broadly classified into two categories: Unsupervised linear clustering algorithms and unsupervised non-linear clustering algorithms. Under Unsupervised linear clustering algorithm category the major algorithms are: k-means clustering algorithm, Fuzzy c-means clustering algorithm, Hierarchical clustering algorithm, Gaussian (EM) clustering algorithm and Quality threshold

clustering algorithm. Under Unsupervised non-linear clustering algorithm category the major algorithms are: MST based clustering algorithm and kernel k-means clustering algorithm.

There are a number of advantages of Density based clustering approach particularly DBSCAN [3][5][6]:

- It does not require you to know the number of clusters in the data a priori, as opposed to k-means.
- It can find arbitrarily shaped clusters.
- The algorithm has a notion of noise.
- It requires just two parameters and is mostly insensitive to the ordering of the points in the database.

Hence, due to the advantages of Density based clustering approach, the proposed work is aimed at: Prediction of Level of Severity of Faults in Software Systems using Density based clustering technique.

The organization of the paper is as follows: section 2 mentions Literature Survey part of the proposed work. The basic things which are taken under consideration while taking the review are various Machine Learning Algorithms which are used by researchers for software fault prediction and related work. Section 3 describes the Methodology part of work done, which shows the steps used in order to reach the objectives and carry out the results and section 4 describes the Results and Discussions part of work done, which shows the actual implementation of the proposed work and also presents the results that are formed while working on the problem. In the last section, Conclusions are deduced on the basis of the results obtained.

2. Literature Survey

The basic hypothesis of software quality prediction is that a module currently under development has defects if a module with the similar product or process metrics in an earlier project (or release) developed in the same environment had defects [7]. Therefore, the information available early within the current project or from the previous project can be used in making predictions. This methodology is very useful for the large-scale projects or projects with multiple releases.

Maintenance managers can apply existing techniques that have been traditionally been used for other types of applications. One system is not enough for prediction purposes. The empirical study detailing software maintenance for web based java applications can be performed to aid in understanding and predicting the software maintenance category and effort [8].

The organizations are using metrics to improve quality and productivity [9] due to the advent of Total Quality Management. Software maintenance organizations are no exception. In 1987, the U.S. Navy established centralized Software Support Activity (SSA) to provide software maintenance for cryptologic systems. At that time two systems were supported and a software maintenance metrics program was established to support the goals of the SSA.

Visual approach [10] can be used to uncover the relationship between evolving software and the way it is affected by software bugs. By visually putting the two aspects close to each other, we can characterize the evolution of software artifacts.

Software maintenance is central to the mission of many organizations. Thus, it is natural for managers to characterize and measure those aspects of products and processes that seem to affect cost, schedule, quality, and functionality of a software maintenance delivery [13]. The importance of software maintenance in today's software industry cannot be overestimated.

Statistical, machine learning, and mixed techniques are widely used in the literature to predict software defects. Khoshgoftaar [14] used zero-inflated Poisson regression to predict the fault-proneness of software systems with a large number of zero response variables. He showed that zero-inflated Poisson regression is better than Poisson regression for software quality modelling. Munson and Khoshgoftaar [15, 16] also investigated the application of multivariate analysis to regression and showed that reducing the number of “independent” factors (attribute set) does not significantly affect the accuracy of software quality prediction.

Menzies, Ammar, Nikora, and Stefano [17] compared decision trees, naïve Bayes, and 1-rule classifier on the NASA software defect data. A clear trend was not observed and different predictors scored better on different data sets. However, their proposed ROCKY classifier outscored all the above predictor models.

Emam, Benlarbi, Goel, and Rai [18] compared different case-based reasoning classifiers and concluded that there is no added advantage in varying the combination of parameters (including varying nearest neighbor and using different weight functions) of the classifier to make the prediction accuracy better.

Bayesian Belief Networks (also known as Belief Networks, Causal Probabilistic Networks, casual Nets, Graphical Probability Networks, Probabilistic Cause-Effect Models, and Probabilistic Influence Diagrams)[19] have attracted much recent attention as a possible solution for the problems of decision support under uncertainty. Although the underlying theory (Bayesian probability) has been around for a long time, the possibility of building and executing realistic models has only been made possible because of recent algorithms and software tools that implement them. Clearly defects are not directly caused by program complexity alone. In reality the propensity to introduce defects will be influenced by many factors unrelated to code or design complexity.

Many modelling techniques have been developed and applied for software quality prediction. These include logistic regression, discriminant analysis [20, 21], the discriminative power techniques, Optimized Set Reduction and artificial neural network [22-23]. For all these software quality models, there is a trade off between the defect detection rate and the overall prediction accuracy. The software quality may be analyzed with limited fault proneness data [24]. An attempt of finding the severity of faults is made by us in [26] and logistic model Trees (LMT) is depicted as the best technique among various machine learning algorithms experimented.

3. Methodology

In course of the research work, the following steps will be required: -

- Study of the metrics needed for prediction of software fault severity.
- Collect the sampled relevant metric data.
- Analyze and refine metrics data.
- The DBSCAN algorithm was first introduced by Ester, et al. [2],

and relies on a density-based notion of clusters. Clusters are identified by looking at the density of points. Regions with a high density of points depict the existence of clusters whereas regions with a low density of points indicate clusters of noise or clusters of outliers. This algorithm is particularly suited to deal with large datasets, with noise, and is able to identify clusters with different sizes and shapes DBSCAN requires two parameters: ϵ (eps) and the minimum number of points required to form a cluster (minPts). It starts with an arbitrary starting point that has not been visited. This point's ϵ -neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized ϵ -environment of a different point and hence be made part of a cluster [3].

If a point is found to be part of a cluster, its ϵ -neighborhood is also part of that cluster. Hence, all points that are found within the ϵ -neighborhood are added, as is their own ϵ -neighborhood. This process continues until the cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. [3]

In a density based clustering a cluster is defined as maximal set of density connected points. The main feature of density based clustering is that it discovers features of arbitrary shape and it can handle noise. Deduce the results on basis of accuracy, precision and recall values.

Pseudocode of the DBSCAN algorithm as described in [3] is given as below:

DBSCAN (D, eps, MinPts)

C = 0

for each unvisited point P in dataset D

mark P as visited

N = getNeighbors (P, eps)

if sizeof(N) < MinPts

```

    mark P as NOISE
else
    C = next cluster
    expandCluster(P, N, C, eps, MinPts)

expandCluster(P, N, C, eps, MinPts)
add P to cluster C
for each point P' in N
    if P' is not visited
        mark P' as visited
        N' = getNeighbors(P', eps)
        if sizeof(N') >= MinPts
            N = N joined with N'
    if P' is not yet member of any cluster
        add P' to cluster C

```

The performance criterion taken is Accuracy percentage means the percentage of the predicted values that match with the expected values for the given data. The desired system is that having the high Accuracy value.

4. Implementation and Results

The first step is to find the structural code and design attributes of software systems i.e. software metrics. The real-time defect data set named as KC3 used in the present work has been accessed from the NASA's MDP (Metric Data Program) data repository. After performing the join operations the Software fault severity dataset is created. There are 21 attributes in the study:

- LOC_BLANK
- BRANCH_COUNT
- LOC_CODE_AND_COMMENT
- LOC_COMMENTS
- CYCLOMATIC_COMPLEXITY
- DESIGN_COMPLEXITY
- ESSENTIAL_COMPLEXITY
- LOC_EXECUTABLE
- HALSTEAD_CONTENT
- HALSTEAD_DIFFICULTY
- HALSTEAD_EFFORT
- HALSTEAD_ERROR_EST
- HALSTEAD_LENGTH
- HALSTEAD_LEVEL
- HALSTEAD_PROG_TIME
- HALSTEAD_VOLUME
- NUM_OPERANDS
- NUM_OPERATORS
- NUM_UNIQUE_OPERANDS
- NUM_UNIQUE_OPERATORS

- LOC_TOTAL

| | |
|-----------------------|--|
| database_Type | weka.clusterers.forOPTICSAndDBScan.Databases.SequentialDatabase |
| database_distanceType | weka.clusterers.forOPTICSAndDBScan.DataObjects.EuclidianDataObject |
| epsilon | 0.9 |
| minPoints | 6 |

Fig. 1: Snapshot of the Parameters Set for DBSCAN

The following are the parameters used in the DBSCAN algorithm implementation in the WEKA (as shown in fig. 1):

- database_Type – It is the used database name and path
- database_distanceType – It tells us the type of distance used. It is set to Euclidian Distance.
- epsilon – It is radius of the epsilon-range-queries. It is set to 0.9 value.
- minPoints – This parameter tells the minimum number of DataObjects required in an epsilon-range-query. It is set to 6 value.

During the clustering the total Clustered Instances are 25 in number and unclustered instances are found to be equal to 4. When the clustered class of the instances is compared with the expected or actual class of the severity of the fault then the incorrectly clustered instances are found to be equal to 4 means the Accuracy percentage is equal to 13.7931 %. Hence, the accuracy is found to be 86.2069%.

The performance of the adaptive Neuro-Fuzzy system i.e. ANFIS is also experimented for the software fault severity dataset of KC3 software. The details of the sugeno based adaptive Neuro-Fuzzy inference system are given below:

- Number of nodes: 156
- Number of linear parameters: 66
- Number of nonlinear parameters: 126
- Total number of parameters: 192
- Number of training data pairs: 29
- Number of checking data pairs: 0
- Number of fuzzy rules: 3

Here also, the accuracy percentage is found to be equal to 86.2069%.

5. Conclusion

In this paper, the performance of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is explored for prediction of level of severity of faults in the object oriented software systems. In this study, NASA's public domain defect dataset named as KC3, written in Java programming language, is used. Here, the Neuro-Fuzzy based inference system named as ANFIS is also experimented for the same dataset for the comparison. The results show that percentage accuracy of DBSCAN and Neuro-Fuzzy based system is same, i.e. 86.2069%, for identifying the level of severity of faults present in the Java based object oriented software systems. Therefore, it is concluded the prediction of different level of severity or impact of faults in object oriented software systems can be done satisfactorily using density based algorithms. Hence, the proposed algorithm can be used to identify modules that have major faults and require immediate attention.

6. References

- [1] Ebru Ardil and Parvinder S. Sandhu, "A soft computing approach for modeling of severity of faults in software systems", International Journal of Physical Sciences Vol. 5 (2), pp. 074-085, February, 2010.
- [2] Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu, "A Density- Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", The Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996.

- [3] <http://en.wikipedia.org/wiki/DBSCAN>
- [4] http://en.wikipedia.org/wiki/Cluster_analysis
- [5] Hans-Peter Kriegel, Martin Pfeifle, "Density-based clustering of uncertain data", Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05), ACM New York, NY, USA ©2005
- [6] Adriano Moreira, Maribel Y. Santos and Sofia Carneiro. "Density-based clustering algorithms – DBSCAN and SNN", Available at: get.dsi.uminho.pt/local/download/SNN&DBSCAN.pdf
- [7] Khoshgoftaar, T. M., Allen, E. B., Ross, F. D., Munikoti, R., Goel, N. & Nandi, A., "Predicting fault-prone modules with case-based reasoning". ISSRE 1997, the Eighth International Symposium on Software Engineering (pp. 27-35), IEEE Computer Society (1997).
- [8] Min-Gu Lee and Theresa L. Jefferson, "An Empirical Study of Software Maintenance of a Web-based Java Application", Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), IEEE (2005).
- [9] Thomas M. Pigoski and Lauren E. Nelson, "Software Maintenance Metrics: A Case Study", Proceedings of IEEE Conference on Software Maintenance, IEEE (1994).
- [10] Marco D' Ambros and Michle Lanza, "Software Bugs and Evolution: A Visual Approach to uncover their relationship", Proceedings of IEEE Conference on Software Maintenance and Reengineering (CSMR' 06), IEEE (2006).
- [11] www.cs.waikato.ac.nz/~ml/weka/
- [12] Rajib Mall, "Fundamentals of software Engineering", Prentice Hall of India, (2001).
- [13] George E. Stark, "Measurements for Managing Software Maintenance", IEEE computer Society (1996).
- [14] Khoshgoftaar, T.M., Gao, K. and Szabo, R.M., "An Application of Zero-Inflated Poisson Regression for Software Fault Prediction", Software Reliability Engineering, ISSRE 2001. Proceedings of 12th International Symposium on, 27-30 Nov. (2001), Page(s): 66 -73.
- [15] Munson, J. and Khoshgoftaar, T., "Regression Modeling of Software Quality: An Empirical Investigation", Information and Software Technology, Volume: 32 Issue: 2, (1990), Page(s): 106 - 114.
- [16] Khoshgoftaar, T.M. and Munson, J.C., "Predicting Software Development Errors using Complexity Metrics", Selected Areas in Communications, IEEE Journal on, Volume: 8 Issue: 2, Feb. (1990), Page(s): 253 -261.
- [17] Menzies, T., Ammar, K., Nikora, A., and Stefano, S., "How Simple is Software Defect Prediction?" Submitted to Journal of Empirical Software Engineering, October (2003).
- [18] Eman, K., Benlarbi, S., Goel, N., and Rai, S., "Comparing case-based reasoning classifiers for predicting high risk software components", Systems Software, Journal of, Volume: 55 Issue: 3, Nov. (2001), Page(s): 301 – 310.
- [19] Fenton, N.E. and Neil, M., "A critique of software defect prediction models", Software Engineering, IEEE Transactions on, Volume: 25 Issue: 5, Sept.- Oct. (1999), Page(s): 675 -689.
- [20] Hudepohl, J. P., Aud, S. J., Khoshgoftaar, T. M., Allen, E. B. & Mayrand, J. Emerald: "software metrics and models on the desktop". IEEE Software (1996)., 13(5), 56-60.
- [21] Khoshgoftaar, T. M., Allen, E. B., Kalachelvan, K. S. & Goel, N., "Early quality prediction: a case study in telecommunications". IEEE Software (1996), 13(1), 65-71.
- [22] Khoshgoftaar, T. M. and Seliya, N., "Tree-based software quality estimation models for fault prediction", METRICS 2002, the Eighth IEEE Symposium on Software Metrics (pp. 203-214). IEEE Computer Society (2002).
- [23] Seliya N., Khoshgoftaar, T.M., Zhong S., "Analyzing software quality with limited fault-proneness defect data", Ninth IEEE international Symposium on 12-14 Oct, (2005).
- [24] Munson, J. C. & Khoshgoftaar, T. M., "The detection of fault-prone programs", IEEE Transactions on Software Engineering, (1992), 18(5), 423-433.
- [25] Sandhu, Parvinder S., Malhotra U., Ardil, E., "Predicting the Impact of the Defect on the Overall Environment in Function Based Systems", World Academy Of Science, Engineering And Technology, issue 56, Aug 2009, pp. 140-43.

- [26] Sandhu, Parvinder Singh, Sunil Kumar and Hardeep Singh, (2007), “Intelligence System for Software Maintenance Severity Prediction”, Journal of Computer Science, Vol. 3 (5), pp. 281-288, 2007.