International Journal of Soft Computing 11 (3): 203-206, 2016
ISSN: 1816-9503

Software Defect Prediction Using Euclidean Distance Probability

Akarsh Goyal, Neel Sheth and N Sujith Kumar Reddy
School of Computer Science and Engineering, Vellore
Institute of Technology University, 632014 Vellore, India

Abstract: In this study we have analyzed basics of software measurements which is indicated by software defect prediction. We have primarily focused on static code measures to find the probability of defect. These measures have been used to estimate the accuracy of the algorithms so that we could predict defect, probability of detection and false alarm based on the PROMISE Software Engineering Repository data set. To do this we have applied k-means clustering and Euclidean distance techniques for probability. This analysis helps us to estimate which attribute when chosen alone give more accuracy then others so that they can be used further for more rigorous assessment by some other means.

Key words: Software defect prediction, McCabe attributes, halstead attributes, probability of defect, data set

INTRODUCTION

A software measurement is a standard of measure of a degree to which asoftwaresystem or process possesses some property. It helps indicate quality of the product, assess the efficiency and productivity of the developers, determine how good our product is compared to other existing solutions or indicate the weakness in our system. Also it helps us determine how well liked our product is or how popular the product is based on customer base. Software measurements aids in analyzing various processing areas which can help in better productivity and profitability like overhead costing, development costing, development time, budget, implementation time, delays, marketing costs etc. Also through this we take into consideration various product related information like user-friendliness, ease of use, design issues, customer satisfaction and popularity which helps in determining how effective is our product or whether the time and money were worth the investment.

Our study will be focused on Software defect Prediction (Menzies and Stefano, 2004) as defects in a software are major concern in the industry. We will be using McCabe features which states that code with complicated pathways are more error-prone and Halstead features that states that code that is hard to read is more likely to be fault prone. We will be using static code measures as they can be automatically and cheaply collected and are widely used to guide software quality predictions.

Defect detectors (Menzies *et al*., 2004) based on staticmeasures are best viewed as probabilistic statements such that the frequency of faults tends to increase in code modules that trigger the detectors. And several large government software contractors won't review software modules unless tools like McCabe predict that they are fault prone. Hence, defect detectors have a major economic impact when they may force programmers to rewrite code.

For doing the Software defect Prediction, we will be applying the concept of K-means in neural networks which will help us to identify various cluster centers so that the software can be classified as having a defect or no defect. Based on our result we will calculate the accuracy percentage. The accuracy will help us in determining which all attributes are more useful in classifying the various software. Also the found out information can be used in the future to make the system more precise by using back propagation network on the training dataset.

Many study have discussed this problem of software defect prediction in detail and also found out methods to reduce this growing issue with software. Some of these study refer to these defects as blind spots left behind while assessing the quality of the major software products. Mostly they have worked on the Promise dataset made public by NASA. regarding their various software. In other study, they have used techniques like data mining and genetic algorithms to predict defects. But the approach used by us is not so much complex and also the accuracy we get out of it is decent as compared to earlier methods.

Corresponding Author: Akarsh Goyal, School of Computer Science and Engineering, Vellore Institute of Technology University, 632014 Vellore, India

Background: This study takes into account static code complexities such as McCabe, Halstead and Lines of Code attributes. Throughout the study we will come across terms given above and K-means clustering, probability calculation using Euclidean distance and accuracy of prediction. So let us define these terms:

Static code analysis: Static program analysis is the analysis of computer software that is performed without actually executing programs .In most cases the analysis is performed on some version of the source code and in the other cases, some form of the object code. It is usually applied to the analysis performed by an automated tool with human analysis being called program understanding, program comprehension or code review. It is used in the verification of properties of software used in safety-critical computer systems and locating potentially vulnerable code.

McCabe attributes: McCabe attributes uses a vast number of software metrics to get the most precise assessment of your application's quality, security and testing. These metrics have been defined below:

- Cyclomatic Complexity Metric (v(G))-Cyclomatic Complexity (v(G)) is a measure of the complexity of a module's decision structure. It is the number of linearly independent paths and therefore, the minimum number of paths that should be tested
- Essential Complexity Metric (ev(G))-Essential Complexity (ev(G)) is measure of the degree to which a module contains unstructured construct. This metric measures the degree of structuredness and the quality of the code. It is used to predict the maintenance effort and to help in themodularization process
- Design ComplexityMetric (S0)-It measures the amount of interaction betweenmodules in a system

Halstead attributes: Halstead features states that code that is hard to read is more likely to be fault prone. Halstead refers to n1* and n2* as the minimum possible number of operators and operands for a module and a program, respectively. This minimum number would be embodied in the programming language itself, in which the required operation would already exist possibly as a function.

K-means clustering: K-means clustering is a method of vector quantization, originally from signal processing. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The cluster center is found out for each cluster. Then each observation is compared with the cluster center using Euclidean distance method to determine the cluster in which they belong. The method is repeated until the cluster centers stop changing.

Probability using euclidean distance: This is done by normalizing the distance between two cluster centers to 0-1. Then the distance between a point and the center is calculated. This distance is multiplied by 100 to get the reverse probability. The value is subtracted from 100 to get the correct probability of the point to lie in a specific cluster.

MATERIALS AND METHODS

Algorithm: Dataset Used: JM1 promise dataset (Menzies, 2004). Data comes from McCabe and Halstead features extractors of source code. These metrics are widely used for defect prediction purposes and software quality.

- Import the dataset into R studio and divide it into training and test datasets. Take first 2000 records for training and use rest of them for testing purpose
- Divide the training dataset into two subsets: one having attribute defects = "Y" (records that have defects) and other having defects = "N" (records not having defects)

R code: results<-subset(promise_s, V22 = "Y")
results<-subset(promise_s, V22 = "N")

- Now apply kmeans clustering on results to obtain cluster means. R code: cluster1 <- kmeans(results,1)
- Cluster 1 details (Fig. 1)
- Now apply kmeans clustering on results to obtain cluster means

R code: cluster2<-kmeans(results,1)

- Cluster 2 details (Fig. 2)
- Now using this cluster centers, we compute Euclidean distance to calculate probability of defect. (Using C language)
- We read each record from testing data set and calculate the probability of defect

```
K-means clustering with 1 clusters of sizes 448

Cluster means:
       V1       V2       V3       V4       V5       V6       V7
1 9.609375 19.65848 1.261161 6.111607 10.99777 6.299107 5.236607
       V8       V9      V10      V11      V12      V13       V14
1 53.51786 54.06598 22.89366 70799.94 0.497433 237.7411 0.09316964
      V15      V16      V17      V18      V19      V20      V21
1 3933.33 1492.672 97.12054 140.6205 32.79241 16.64062 72.79911
```

Fig. 1: Screenshot of cluster 1

```
> cluster2
K-means clustering with 1 clusters of sizes 1552

Cluster means:
        V1       V2        V3       V4       V5       V6      V7
1 4.936856 9.610825 0.3427835 2.905928 5.421392 3.431057 3.05799
        V8       V9      V10      V11       V12      V13       V14
1 25.93814 32.5352 16.28878 21559.19 0.2077062 113.5013 0.1169523
       V15      V16      V17      V18      V19      V20      V21
1 1197.733 623.5083 46.00773 67.49356 17.78673 12.96778 36.39626
```

Fig. 2: Screenshot of cluster 2

- Based on the probability of defect, we calculate the no. of records belonging to each range of probability, no. of records having defects in the given range of probability and no. of records not having defects in the given range of probability
- Based on no. of records having defects and not having defects we calculate the accuracy of our prediction for each range of probability
- Now using the same procedure we calculate the accuracy of the prediction using only Halstead and only McCabe attributes from the dataset

RESULTS AND DISCUSSION

We devised our own method to predict the probability of the software having defect based on the range of outputs.

Table 1 shows our accuracy results for the dataset based on distance prediction method (Table 1). Based on the analysis of JM1 promise data set while considering all the attributes, only Halstead attributes and only McCabe attributes we found following details.

- In the range of 100-30% the prediction of Probability of defect is accurate by 40-45%
- For probability of not having a defect greater than 70%the prediction is 78% accurate

When considering Halstead attributes:

- In the range of 100-30% the prediction of probability of defect is accurate by 41-44%
- For probability of not having a defect >70% the prediction is 78% accurate
- When considering McCabe attributes
- For probability of having a defect greater than 90% the prediction is 52% accurate
- In the range of 90-30% the probability of defect is accurate by 31-38%
- For probability of not having defect ranging from 10-50% the prediction is 60-69% accurate
- For probability of not having defect ranging from 50-100% the prediction is 70-78% accurate

Table 1: Results

	Probability ranges									
Parameters	100-90	90-80	80-70	70-60	60-50	50-40	40-30	30-20	20-10	10-0
All Attributes										
No. of records	49	68	77	124	205	50	57	2149	907	314
No. of records with defects	21	31	28	55	87	24	15	327	196	76
No. of records without defects	28	37	49	69	118	26	42	1822	711	238
Accuracy of having defects (%)	42.8571	44.444	41.237	42.452	42.447	42.913	41.428	21.158	21.269	21.5
Accuracy of not having defects (%)	57.1429	55.555	58.762	57.547	57.552	57.068	58.571	78.841	78.730	78.5
Halstead Attributes										
No. of records	49	68	77	124	205	50	57	2149	907	314
No. of records with defects	21	31	28	55	87	24	15	327	196	76
No. of records without defects	28	37	49	69	118	26	42	1822	711	238
Accuracy of having defects (%)	42.8571	44.444	41.237	42.452	42.447	42.913	41.428	21.158	21.269	21.5
Accuracy of not having defects (%)	57.1429	55.555	58.762	57.547	57.552	57.068	58.571	78.841	78.730	78.5
McCabe Attributes										
No. of records	19	122	212	272	284	74	581	1537	715	185
No. of records with defects	10	40	61	116	120	16	90	225	145	37
No. of records without defects	9	82	151	156	164	58	491	1312	569	148
Accuracy of having defects (%)	52.6316	35.461	31.444	36.32	38.173	36.927	28.964	21.863	21.572	21.5
Accuracy of not having defects (%)	47.368	64.539	68.555	63.68	61.826	63.072	71.035	78.136	78.42	778.5

CONCLUSION

Using the described prediction method, it becomes relatively easy to predict whether the given software has a defect or not based on its static code attributes. The method is simple to implement and can be efficiently used to form a preliminary opinion on whether to conduct an exhaustive testing of software to find defects or how much resources to allocate for testing based on the probability of the defect. Also the prescribed method has very high accuracy of nearly 80% in determining whether the software does not have a defect and thus can be used for Software Defect Prediction.

REFERENCES

Menzies, T. and J.S. di Stefano, 2004. How good is your blind spot sampling policy. Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering, March 25-26, 2004, Tampa, FL., USA., pp: 129-138.

Menzies, T., 2004. JM1/software defect prediction. NASA Metrics Data Program, December 2, 2004.

Menzies, T., J.S. di Stefano, A. Orrego and R. Chapman, 2004. Assessing predictors of software defects. Proceedings of the Workshop on Predictive Software Models, September 2004, Chicago, IL., USA -.