IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 3, Issue 6, January 2015)

PSO Optimized Software Fault Prediction system using Fuzzy C-Means

Dazy Arya
M. Tech. Scholar, Computer Science Dept.
Rajasthan College of Engineering for Women,
Jaipur India
dazyarya@gmail.com

Dr. Rajeev Yadav
Principal, Rajasthan College of Engineering for Women,
Jaipur India
yadavrajeev6@gmail.com

Abstract –**Early detection of fault prone software components enables verification experts to concentrate their time and resources on the problem areas of the software systems under development. In this paper, performance comparison of a Software Fault Prediction System using Fuzzy c-means clustering approach and a hybrid technique (Combination of Fuzzy c-means and Particle Swarm Optimization) a has been performed with the real time data set named PC1, taken from NASA MDP software projects. The performance is recorded on the basis of accuracy, reliability, RMSE and MAE values.**

Keywords – **Fault-Proneness, Fuzzy C-Means, Particle Swarm Optimization, NASA MDP, etc.**

I. Introduction

Faults are major problem in software systems that need to be resolved. Fault is a flaw that results in failure. We should have to know the clear difference between bug, fault and failure. Failure is deviation of software actions from the expected outcomes. A fault in software is a flaw that results in failure. Bug occurs when specified requirements of the software do not conform. There are many number of software having number of faults are delivered to the market [1].

A fault is a defect, an error in source code that causes failures when executed. A fault prone software module is the one containing more number of expected faults. Accurate prediction of fault prone modules enables the verification and validation activities focused on the critical software components.

A software fault is a defect that causes software failure in an executable product. For each execution of the software program where the output is incorrect, we observe a failure. Software engineers distinguish software faults from software failures. Faults in software systems continue to be a major problem. Various systems are delivered to users with excessive faults. This is despite a huge amount of development effort going into fault reduction in terms of quality control and testing. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. An inadequate amount of valuable work in this area has been carried out previously. Regardless of this it is difficult to identify a reliable approach to identifying fault-prone software components. Using software complexity measures, the techniques build models, which categorize components as likely to contain faults or not.

Till now there are proposed numerous methods for data clustering methods. The algorithms provide a satisfying measure for the classification and mining of data. The software fault prediction is also now using the data clustering techniques because of the features and the functions they are expected to deliver. The clustering techniques till now have solved many purposes yet the satisfying result could not be guaranteed. In this research work, we have tried to modify the previous algorithms for the better results. We do not say that it is the end of research in this segment but it will definitely provide the new researchers with the scope to bring new considerations that could serve the future demands.

The main objective of this paper is to design a Software Fault Prediction System using Fuzzy c-means clustering approach and a hybrid technique (combination of Fuzzy c-means and Particle Swarm Optimization). The results after classification of software fault data come in terms of certain efficiency parameters like Accuracy, Reliability, Mean Absolute Error, and Root Mean Squared Error in order to compare both approaches.

II. Proposed Methodology

1. Find the structural code and requirement attributes

The first step is to find the structural code and requirement attributes of software systems i.e. software metrics. The real time defect data sets are taken from the NASA's MDP (Metric Data Program) data repository, [online] Available: http://mdp.ivv.nasa.gov.innamed as PC1 dataset which is collected from a flight software from an earth orbiting satellite coded in C programming language, containing 1107 modules and only 109 have their requirements specified. PC1 has 320 requirements available and all of them are associated with program modules. All these data sets varied in the percentage of defect modules, with the PC1 dataset containing the least number of defect modules.

2. Select the suitable metric values as representation of statement

The Suitable metric values used are fault and without fault attributes, we set these values in database create in MATLAB R2010 A as 0 and 1. Means 0 for data with fault and 1 for data without fault. The metrics in these datasets (NASA MDP dataset) describe projects which vary in size and complexity, programming languages, development processes, etc. When reporting a fault prediction modelling experiment, it is important to describe the characteristics of the datasets. Each data set contains twenty-one software metrics, which describe product's size, complexity and some structural properties. We use only fault and without attributes to classify the selected NASA MDP PC1 dataset. Also the product metrics and product module metrics available in dataset which can also be use are the product requirement metrics are as follows:

- Module
- Action
- Conditional
- Continuance
- Imperative
- Option
- Risk_Level
- Source
- Weak_Phrase

The product module metrics are as follows:

1. Module
2. Loc_Blank
3. Branch_Count
4. Call_Pairs
5. LOC_Code_and_Comment
6. LOC_Comments
7. Condition_Count
8. Cyclomatic_complexity
9. Cyclomatic_Density
10. Decision_Count
11. Edge_Count
12. Essential_Complexity
13. Essential_Density
14. LOC_Executable
15. Parameter_Count
16. Global_Data_Complexity
17. Global_Data_Density
18. Halstead_Content
19. Halstead_Difficulty
20. Halstead_Effort
21. Halstead_Error_EST
22. Halstead_Length
23. Halstead_Prog_Time
24. Halstead_Volume
25. Normalized_Cyclomatic_Complexity
26. Num_Operands
27. Num_Operators
28. Num_Unique_Operands
29. Num_Unique_Operators
30. Number_Of_Lines
31. Pathological_Complexity
32. LOC_Total

Figure 1 and 2 show flow diagrams for Fuzzy c-means clustering approach and hybrid approach respectively.

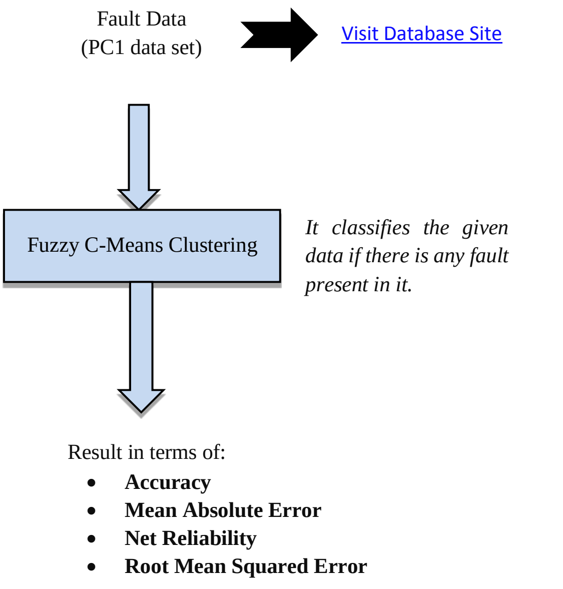


Figure 1: Flow diagram for Fuzzy C-means clustering Approach

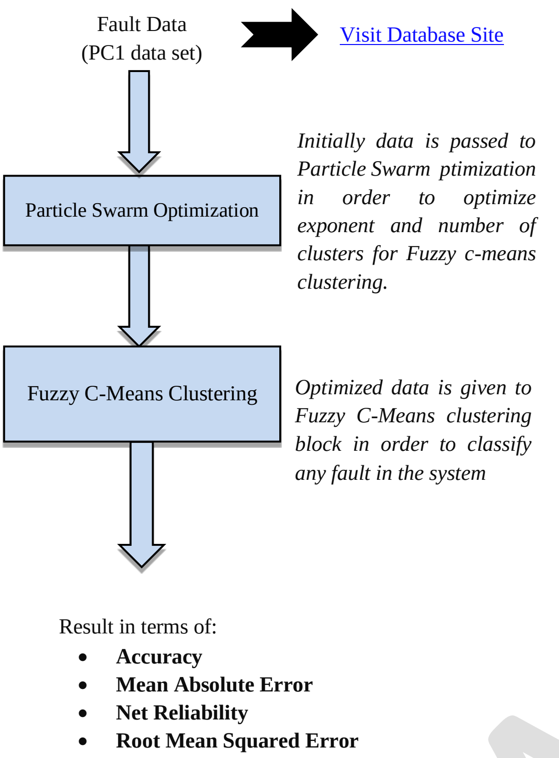

Fault Data
(PC1 data set)
Visit Database Site
Particle Swarm Optimization
Initially data is passed to Particle Swarm ptimization in order to optimize exponent and number of clusters for Fuzzy c-means clustering.
Fuzzy C-Means Clustering
Optimized data is given to Fuzzy C-Means clustering block in order to classify any fault in the system
Result in terms of:
Accuracy
Mean Absolute Error
Net Reliability
Root Mean Squared Error

Figure 2: Flow diagram for hybrid (PSO-Fuzzy C-Means) technique

In this paper we have developed a software fault prediction module using two methods:

- Fuzzy c-means clustering (FCM) approach.
- Hybrid technique (combination of Fuzzy c-means and Particle Swarm Optimization).

PC1 software fault database is used available at NASA's research website. In the first method Fuzzy c-means clustering approach is used to detect any fault present in the data. In the hybrid approach, PSO optimizes the 'exponent' and 'number of clusters' for FCM approach and then FCM classifies the dataset in order to make an efficient and supervised classification model.

Fuzzy C-Means Clustering

Fuzzy C-Means iteratively moves the cluster centers to the "right" location within a data set. Objective function based fuzzy clustering algorithms such as the fuzzy c-means (FCM) algorithm has been used extensively for different tasks such as pattern recognition, data mining, and image processing and fuzzy modeling.

In general, cluster analysis refers to a broad spectrum of methods which try to subdivide a data set X into c subsets (clusters) which are pairwise disjoint, all nonempty, and reproduce X through union. The clusters then are termed a hard (i.e., non-fuzzy) c-partition of X.

Parameters of the FCM Algorithm

Number of Clusters: The number of clusters c is the most important parameter, in the sense that the remaining parameters have less influence on the resulting partition. When clustering real data without any a priori information about the structures in the data, one usually has to make assumptions about the number of underlying clusters. The chosen clustering algorithm then searches for c clusters, regardless of whether they are really present in the data or not. Two main approaches to determining the appropriate number of clusters in data can be distinguished:

A. *Validity measures:* Validity measures are scalar indices that assess the goodness of the obtained partition. Clustering algorithms generally aim at locating wellseparated and compact clusters. When the number of clusters is chosen equal to the number of groups that actually exist in the data, it can be expected that the clustering algorithm will identify them correctly. When this is not the case, misclassifications appear, and the clusters are not likely to be well separated and compact. Hence, most cluster validity measures are designed to quantify the separation and the compactness of the clusters.

B. *Iterative merging or insertion of clusters:* The basic idea of cluster merging is to start with a sufficiently large number of clusters, and successively reduce this number by merging clusters.

Fuzziness Parameter: The weighting exponent m is a rather important parameter as well, because it significantly influences the fuzziness of the resulting partition.

Termination Criterion: The FCM algorithm stops iterating when the norm of the difference between U in two successive iterations is smaller than the termination parameter ϵ. For the maximum norm $max_{ik}(|\mu_{ik}^{(l)} - \mu_{ik}^{(l-1)}|)$. The usual choice is $\epsilon = 0.001$, even though $\epsilon = 0.01$ works well in most cases, while drastically reducing the computing times.

Norm-Inducing Matrix: The shape of the clusters is determined by the choice of the matrix A in the distance measure. A common choice is $A = I$, which gives the standard Euclidean norm:

$$D_{ik}^2 = (z_k - v_i)^T (z_k - v_i) \qquad (1)$$

Where v_i are ordinary means of the clusters.

Let $\{\mathrm{x}_1, \mathrm{x}_2, .., \mathrm{x}_N\}$ be a set of N data objects represented by n-dimensional feature vectors.

$$\mathrm{x}_k = [x_{1k}, \dots \dots, x_{nk}]^T \in R^n \qquad (2)$$

A set of N feature vectors is then denoted as a data matrix of $n \times N$.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nN} \end{bmatrix} \qquad (3)$$

A fuzzy clustering algorithm partitions the data X into M fuzzy clusters, forming a fuzzy partitioning. A fuzzy partition can be conveniently represented as a matrix, U, whose elements $u_{ik} \in [0;\ 1]$ represents the membership degree of x_k in cluster $'i'$. Hence, the i^{th} row of U contains values of the i^{th} membership function in the fuzzy partition. Objective function based fuzzy clustering algorithms minimize an objective function of the type

$$J(X; U, V) = \sum_{i=1}^{M} \sum_{k=1}^{N} (u_{ik})^m d^2(\mathrm{x}_k, \mathrm{v}_i) \quad (4)$$

Where,

$$V = [\mathrm{v}_1, \dots, \mathrm{v}_M]^T \in \mathbb{R}^n \qquad (5)$$

is an M-tuple of cluster prototypes which have to be determined, and $m \in (1;\ \infty)$ is a weighting exponent which determines the fuzziness of the clusters in order to avoid the trivial solution, constraints must be forced on U.

$$\sum_{i=1}^{M} u_{ik} = 1, \forall k \qquad (6)$$

$$0 < \sum_{i=1}^{N} u_{ik} < N, \forall i \qquad (7)$$

These constraints imply that the sum of each column of U is 1. Further, there may be no empty clusters, but the distribution of membership among the M fuzzy subsets is not constrained. The prototypes are typically selected to be idealized geometric forms such as linear varieties (e.g. FCV algorithm) or points (e.g. GK or FCM algorithms). When point prototypes are used, the general form of the distance measure is given by

$$d^2(\mathrm{x}_k, \mathrm{v}_i) = (\mathrm{x}_k - \mathrm{v}_i)^T A_i (\mathrm{x}_k - \mathrm{v}_i) \qquad (8)$$

Where the norm matrix A_i is a positive definite symmetric matrix. The FCM algorithm uses the Euclidian distance measure, i.e. $A_i = I \forall_i$, while the GK algorithm uses the Mahalonibisdistance, i.e. $A_i = P_i^{-1}$ with P_i the covariance matrix of cluster i, and the additional volume constraint$|A_i| = \rho_i$.

The FCM algorithms are best described by recasting conditions in matrix-theoretic terms [3]. Towards this end, let U be a real $c \times N$ matrix, $U = [u_{ik}]$. U is the matrix representation of the partition $\{\ Y_i\}$ in the situation

$$u_i(\mathrm{y}_k) = u_{ik} = \begin{Bmatrix} 1; & \mathrm{y}_k \in Y_i \\ 0; & otherwise \end{Bmatrix} \qquad (9)$$

$$\sum_{i=1}^{M} u_{ik} > 0 \quad for\ all\ i \qquad (10)$$

$$\sum_{i=1}^{M} u_{ik} = 1 \quad for\ all\ k \qquad (11)$$

In equation (9), u_i is a function such that: $u_i: Y \to \{0, 1\}$. In conventional models, u_i is the characteristic function of, Y_i: in fact, u_i and Y_i determine one another, so there is no harm in labelling u; the i$^{\text{th}}$ hard subset of the partition (It is unusual, of course, but is important in terms of understanding the term “fuzzy set”). Conditions of equations (10) and (11) are equivalent, so U is termed a hard c-partition of Y. Generalizing this idea, we refer to U as a fuzzy c-partition of Y when the elements of U are numbers in the unit interval [0, 1] that continue to satisfy both equations (10) and (11). The basis for this definition are c functions $u_i: Y \to \{0, 1\}$ whose values $u_i(\mathrm{y}_k) \in [0,1]$ are interpreted as the grades of membership of the y_ks in the “fuzzy subsets” u_i of Y.

Particle Swarm Optimization (PSO)

PSO is a technique used to explore the search space of a given problem to find the settings or parameters required to maximize or minimize a particular objective.

In PSO, a neighbourhood is defined for each individual particle as the subset of particles which it is able to communicate with. The first PSO model used a Euclidian neighbourhood for particle communication, measuring the actual distance between particles to determine which were close enough to be in communication. This was done in imitation of the behaviour of bird flocks, similar to biological models where individual birds are only able to communicate with other individuals in the immediate vicinity. The Euclidian neighbourhood model was abandoned in favour of less computationally intensive models when research focus was shifted from biological modelling to mathematical optimization. Topological neighbourhoods unrelated to the locality of the particle came into use, including what has come to be recognized as a global neighbourhood, *gbest* model, where each particle is associated with and able to obtain information from every other particle in the swarm.

Particle Swarm Algorithm

```
1.  Begin
2.  Factor settings and swarm initialization
3.  Evaluation
4.  g = 1
5.  While (the stopping criterion is not met) do
6.  for each particle
7.  Update velocity
8.  revise place and localized best place
9.  Evaluation
10. End For
11. Update leader (global best particle)
12. g + +
13. End While
14. End
```

The PSO procedure has various phases consist of Initialization, Evaluation, Update Velocity and Update Position. Equation (12) is used for updating the velocity:

$$v_1(t) = \underbrace{\overline{wv_l(t-1)}}_{inertia} + \underbrace{c_1r_1\left(x_1^{\#}(t-1) - \overline{x_l(t-1)}\right)}_{Personal\ influence} + \underbrace{c_2r_2\left(x^{*}(t-1) - \overline{x_l(t-1)}\right)}_{Social\ influence} \quad (12)$$

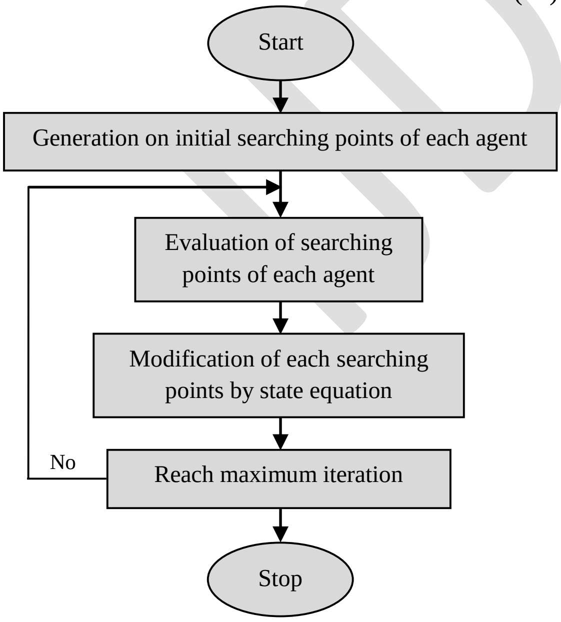
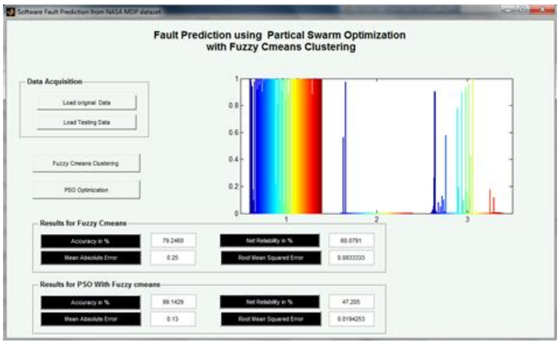

Start
Generation on initial searching points of each agent
Evaluation of searching points of each agent
Modification of each searching points by state equation
Reach maximum iteration
No
Stop

Figure 3: Flow chart of PSO

III. Simulation and Results

Simulation is carried out using MATLAB 2010a:


Software Fault Prediction from NASA MDP dataset
Fault Prediction using Partical Swarm Optimization
with Fuzzy Cmeans Clustering
Data Acquisition
Load original Data
Load Testing Data
Fuzzy Cmeans Clustering
PSO Optimization
1
0.8
0.6
0.4
0.2
0
1
2
3

Results for Fuzzy Cmeans

Accuracy in %	79.2468	Net Reliability in %	60.0791
Mean Absolute Error	0.25	Root Mean Squared Error	0.0833333

Results for PSO With Fuzzy cmeans

Accuracy in %	99.1429	Net Reliability in %	47.205
Mean Absolute Error	0.13	Root Mean Squared Error	0.0194253

Figure 4: Graphical User Interface (GUI) for proposed work

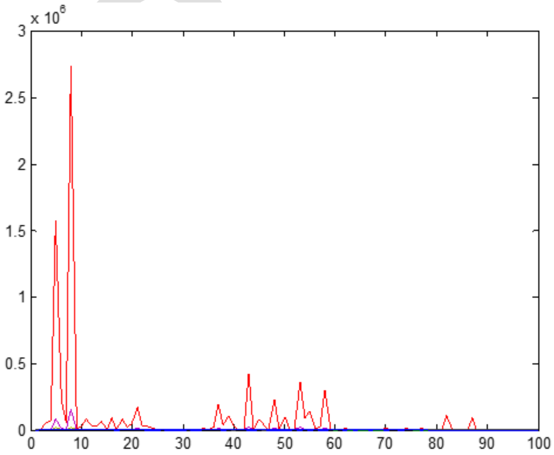


Figure 5: Input PC1 dataset with attributes (fault and without fault)

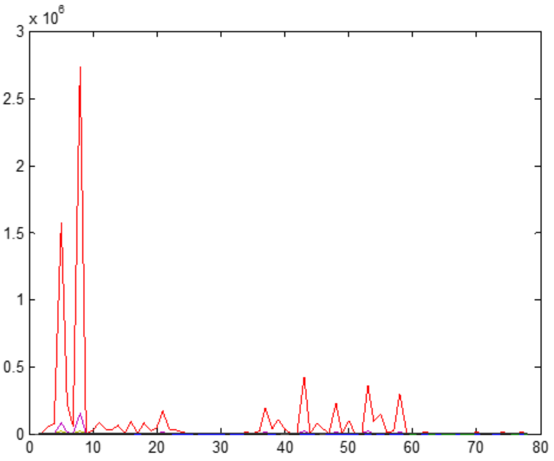


Figure 6: Input PC1 dataset with fault attributes when separating fault attributes from input data

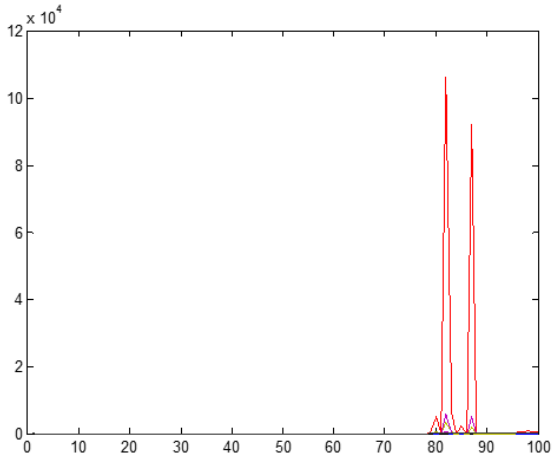


Figure 7: Input PC1 dataset with without fault attributes when separating without fault attributes from input data

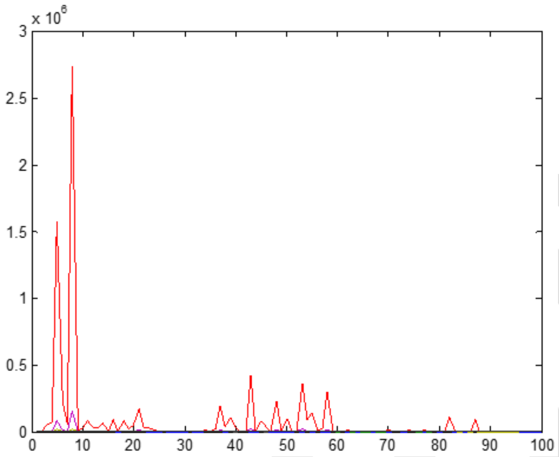


Figure 8: Data Input (PC1 database without attributes)

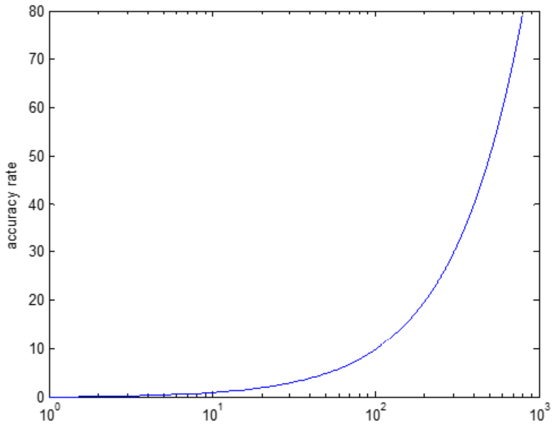


Figure 9: Accuracy graph with FCM approach

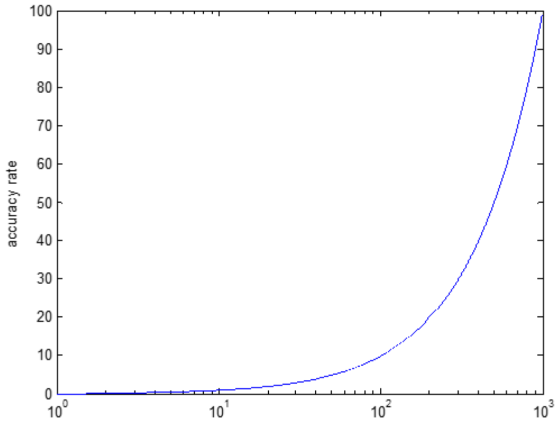


Figure 10: Accuracy graph with PSO-FCM based hybrid approach

Table 1: Performance comparison for Fuzzy C-means and Hybrid (PSO-Fuzzy C-means) technique

Evaluation Parameter	Fuzzy c-means Approach	Hybrid (PSO-FCM) Approach
Accuracy	79.24	99.14
Net Reliability	60.07	47.20
Mean Absolute Error (MAE)	0.25	0.13
Root Mean Squared Error (RMSE)	0.083	0.019

IV. Conclusion

In this paper, a Software Fault Prediction System is implemented using Fuzzy C-means clustering and hybrid (Fuzzy c-means + PSO) techniques. Fuzzy clustering based techniques are discussed for the comparative analysis in order to predict level of impact of faults in NASA's public domain defect dataset. Predicting faults in the software life cycle can be used to improve software process control and achieve high software reliability. It was found that the hybrid method gives more accuracy and less errors as compared to Fuzzy C-means clustering method on the basis of evaluation parameters: accuracy, reliability, MSE and RMSE.

References

[1] Jaakkola T., and Haussler D., "Exploiting generative models in discriminative classifiers", In Advances in Neural Information Processing Systems 1, MIT Press, pp. 487–493, 1998.

[2] Kazama J., and Tsujii J., "Evaluation and extension of maximum entropy models with in equality constraints", Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP2003), pp. 137–144, 2003.

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary research
Website: www.ijdacr.com (Volume 3, Issue 6, January 2015)

[3] Anderberg M. R., Cluster Analysis for Applications, Academic Press, New York, 1973.

[4] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. IEEE Transactions on Software Engineering, 26(8):797–814, 2000.

[5] Michael R. Anderberg. Cluster analysis for applications. Academic Press, 1973.

[6] Section 13.4 of Mardia et al., 1979, and Section 2 of Veltkamp and Hagedoorn, 2000

[7] Sections 13.4 and 14.2.3 of Mardia et al., 1979

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, " A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).

[9] Nam Hun Park Won Suk Lee, "Statistical Grid-based Clustering over Data Streams". SIGMOD Record, Vol. 33, No. 1, March 2004.

[10] Paul S. Bradley, Usama Fayyad, and Cory Reina. Scaling EM (Expectation-Maximization) Clustering to Large Databases. Technical Report MSR-TR-98-35, Microsoft Research, Redmond, WA, USA, October 1999.

[11] Binu Thomas, Raju G., and Sonam Wangmo, “A Modified Fuzzy C-Means Algorithm for Natural Data Exploration” World Academy of Science, Engineering and Technology 25 2009.

[12] Bose S.K., “Presenting a Novel Neural Network Architecture for Membrane Protein Prediction”, Intelligent Engineering Systems, INES’06, Proceedings. International Conference, 2006.

[13] Yuan Chen, “Research on software defect prediction based on data mining”, IEEE 2nd International Conference on Computer and Automation Engineering (ICCAE), 2010.

[14] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. IEEE Transactions on Software Engineering, 26(8):797–814, 2000.

[15] N. E. Fenton and M. Neil. A critique of software defect prediction models. IEEE, Transactions on Software Engineering, 25(5):675–689, 1999.

[16] Pei Wang, “Software Defect Prediction Scheme Based on Feature Selection”, IEEE, International Symposium on Information Science and Engineering (ISISE), 2012.

[17] Qinbao Song, “Software defect association mining and defect correction effort prediction”, IEEE Transactions on Software Engineering, Vol. 32, Issue: 2, February 2006.

[18] Lanubile F., Lonigro A., and Visaggio G., “Comparing Models for Identifying Fault-Prone Software Components”, Proceedings of Seventh International Conference on Software Engineering and Knowledge Engineering, 1995, pp. 12-19, 1995.

[19] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl, S.J. Aud, “Application of neural networks to software quality modeling of a very large telecommunications system”, IEEE Transactions on Neural Networks, 8(4), pp. 902-909, 1997.

[20] Saida Benlarbi, Khaled El Emam, Nishith Geol, “Issues in Validating Object-Oriented Metrics for Early Risk Prediction”, by Cistel Technology, Ontario Canada, 1999.

[21] Runeson, Claes Wohlin, Magnus C. Ohlsson, “A Proposal for Comparison of Models for Identification of FaultProneness”, Dept. of Communication Systems, Lund University, Profes 2001, pp. 341-355, 2001.

[22] Jiang, Y., Cukic, B., Menzies, T., “Fault Prediction Using Early Lifecycle Data”, In the 18th IEEE Symposium on Software Reliability Engineering (ISSRE 2007), IEEE Computer Society, Sweden, pp. 237-246, 2007.

[23] Mahaweerawat A., “Fault-Prediction in object oriented software’s using neural network techniques”, Advanced Virtual and Intelligent Computing Center (AVIC), Department of Mathematics,Faculty of Science, Chulalongkorn University, Bangkok, Thailand, pp. 1-8, 2004.

[24] Bellini, P., “Comparing Fault-Proneness Estimation Models”, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), vol. 0, 2005, pp. 205-214, 2005.

[25] Nurudeen Sherif, Nurudeen Mohammed, " Using Fuzzy Clustering and Software Metrics to Predict Faults in large Industrial Software Systems" IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, ISSN: 2278-8727, Volume 13, Issue 6, PP 32-36, Jul. - Aug. 2013.

[26] Supreet Kaur, and Dinesh Kumar, "Software Fault Prediction in Object Oriented Software Systems Using Density Based Clustering Approach". International Journal of Research in Engineering and Technology (IJRET), ISSN: 2277-4378, Vol. 1, No. 2 March 2012.

[27] Neeraj Mohan, Parvinder S. Sandhu, and Hardeep Singh, " Impact of Faults in Different Software Systems: A Survey" World Academy of Science, Engineering and Technology, 2009.

[28] Thomas J. Ostrand and Elaine J. Weyuker, "A Tool for Mining Defect Tracking Systems to Predict Fault-Prone Files", 1st international workshop on mining software repositories, pp. 85-89, 2005.

[29] Brian Randell, “Facing Up to Faults”, The Computer Journal, Vol. 43, January 2000.

[30] Supreet Kaur, Dinesh Kumar, “Quality Prediction of Object Oriented Software Using Density Based Clustering Approach”, IACSIT International Journal of Engineering and Technology, Vol.3, No.4, August 2011.

[31] Fuzzy C-means (FCM) algorithm, online available at: http://hayoungkim.tistory.com/20.