Concise Papers

Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm

Partha Sarathi Bishnu and
Vandana Bhattacherjee, *Member*, *IEEE*

Abstract—Unsupervised techniques like clustering may be used for fault prediction in software modules, more so in those cases where fault labels are not available. In this paper a Quad Tree-based K-Means algorithm has been applied for predicting faults in program modules. The aims of this paper are twofold. First, Quad Trees are applied for finding the initial cluster centers to be input to the K-Means Algorithm. An input threshold parameter δ governs the number of initial cluster centers and by varying δ the user can generate desired initial cluster centers. The concept of clustering gain has been used to determine the quality of clusters for evaluation of the Quad Tree-based initialization algorithm as compared to other initialization techniques. The clusters obtained by Quad Tree-based algorithm were found to have maximum gain values. Second, the Quad Tree-based algorithm is applied for predicting faults in program modules. The overall error rates of this prediction approach are compared to other existing algorithms and are found to be better in most of the cases.

Index Terms—K-Means clustering, Quad Tree, software fault prediction.

1 Introduction

K-Means clustering is a nonhierarchical clustering procedure in which items are moved among sets of clusters until the desired set is reached [5]. The partitioning of data set is such that the sum of intracluster distances is reduced to an optimum value [23], [27]. K-Means is simple and a widely used clustering algorithm. However, it has some inherent drawbacks. First, the user has to initialize the number of clusters which is very difficult to identify in most of the cases. Second, it requires selection of the suitable initial cluster centers which is again subject to error. Since the structure of the clusters depends on the initial cluster centers this may result in an inefficient clustering. Third, The K-Means algorithm is very sensitive to noise. In [8], a method using Quad Trees has been proposed as an initialization of K-Means algorithm. The Quad Tree-based method assigns the appropriate initial cluster centers and eliminates the outliers [26] hence overcoming the second and third drawback of K-Means algorithm. In this study, we focus on a practical problem that occurs when the fault data for modules are not available. To solve this challenging problem, researchers have applied a combination of clustering techniques to cluster modules, and this process was followed by an evaluation phase of an expert [2], who was an experienced engineer and labeled each cluster as fault-prone or not fault-prone by examining not only the representative points of each cluster, but also some statistical data such as global mean, median, and percentile of each metric. However, their approach required a human expert during the prediction process and it is not always possible to find an experienced expert who would have the duty to label each cluster. In this paper, the Quad Tree-based K-Means algorithm (QDK) [8] has been applied for predicting faults in program modules. The objectives of this paper are as follows: First, Quad Trees are applied for finding initial cluster centers for K-Means algorithm. By varying the value of threshold parameter δ a user can generate a desired number of cluster centers to be used as input to the simple K-Means algorithm. Second, the Quad Tree-based algorithm is applied for predicting faults in program modules. The overall error rates of this prediction approach are compared to other existing algorithms and are found to be better in most of the cases. Clustering gain values for the best cluster by K-Means and by Quad Tree-based algorithm are very close thereby proving the effectiveness of the algorithm. To compare the performance of QDK for initialization of K-Means, experiments have been conducted in which Quad Tree-based algorithm and two other initialization techniques, Likas et al., Global K-Means algorithm [23], [24] and SAS 2004 [23], [25] have been executed and results are compared on the basis of evaluation parameters. The QDK algorithm performs fairly well on all the parameters. The Global K-Means algorithm considers each data item in each iteration leading to high complexity when number of data items and number of clusters are large and these scalability issues have also been raised by the authors. The SAS 2004 algorithm even though being linear does not provide any guidance regarding the selection of their distance measure [23].

The remaining part of the paper is organized as follows: Section 2 presents the related work on the topic. Section 3 presents an overview on the theory of Quad Tree and the initialization algorithm. Section 4 presents the experimental design. Section 5 presents analysis of the results while Section 6 presents the conclusion.

2 Related Work

Zhong et al. [2], [3] applied clustering techniques and expert-based approach for software fault prediction problem. They applied K-Means and Neural-Gas techniques on different real data sets and then an expert explored the representative module of the cluster and several statistical data in order to label each cluster as fault-prone or not fault-prone. And based on their experience Neural-Gas-based prediction approach performed slightly worse than K-Means clustering-based approach in terms of the overall error rate on large data sets. But their approach is dependent on the availability and capability of the expert. Seliya and Khoshgoftaar [4] proposed a constrained based semi-supervised clustering scheme. They showed that this approach helped the expert in making better estimations as compared to predictions made by an unsupervised learning algorithm. Seliya et al. [12] have proposed a semi-supervised clustering approach for software quality analysis with limited fault-proneness data. Most recently Catal et al. [1] proposed a metric threshold and clustering-based approach for software fault prediction. The results of their study demonstrate the effectiveness of metrics threshold and show that the stand-alone application of metrics threshold is easier than the clustering and metrics thresholds-based (two stage) approach because the selection of number of clusters is performed heuristically in this clustering-based method. In our present study we have presented comparative results performed on same data sets as in [1]. Bhattacherjee and Bishnu have applied unsupervised learning approach for fault prediction in software module in [16], [28]. In their work, the false negative rates (FNR) for the clustering-based approach is less than that for metrics-based approach, while the false positive rates (FPR) are better for the metrics-based approach. The overall error rates for both approaches remain the same. Supervised techniques have however been applied for software fault prediction [13] and software effort prediction [14], [15]. Several methods for initialization of K-Means algorithm are available in literature. Tibshirani et al. suggest a statistical method based on gap statistic to find the optimal number of clusters [20]. Pelleg and Moore suggest an algorithm which efficiently searches the space of cluster locations and number of clusters to optimize the Bayesian Information Criterion and Akaike Information

• *The authors are with the Department of Computer Science and Engineering, Birla Institute of Technology, Extension Center Lalpur, PO Lalpur, Ranchi 834001, Jharkhand, India.*
E-mail: psbishnu@gmail.com, vbhattacherjee@ieee.org.

Manuscript received 31 Mar. 2010; revised 26 Aug. 2010; accepted 3 July 2011; published online 19 July 2011.
Recommended for acceptance by D. Tao.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-03-0193.
Digital Object Identifier no. 10.1109/TKDE.2011.163.

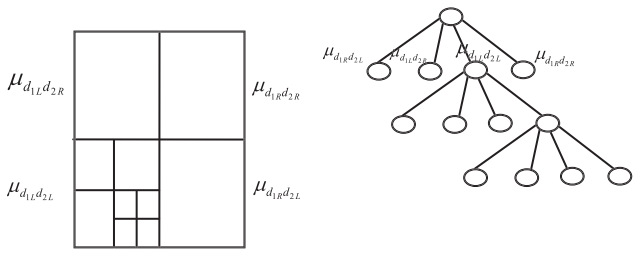

Fig. 1. Quad Tree (for two dimensions).

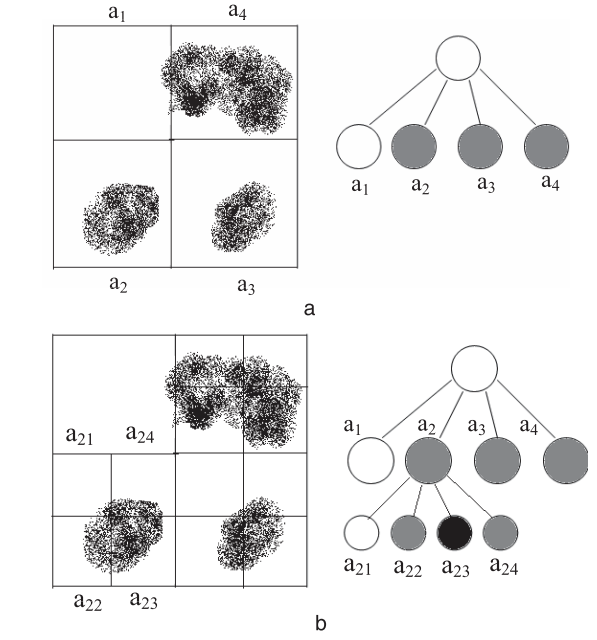

Fig. 2. Quad Tree implementation.

Criterion [10]. Laszlo and Mukherjee present an approach for finding the set of centers by constructing a Hyper-Quad Tree on the set of data. Genetic algorithm has been used for evolving centers in the K-Means algorithm and also for finding a good partitioning [9]. An evaluation of several initialization techniques for K-Means algorithm is presented in [23].

3 Overview of Quad Tree and Proposed Initialization Algorithm

3.1 Quad Tree

A Quad Tree in two dimensional spaces is a 4-way branching tree that represents recursive decomposition of space using separators parallel to the coordinate axis. At each level a square subspace is divided into four equal size squares [17], [18]. This data structure was named a Quad Tree by Finkel and Bentley in 1974 [19]. The definition of a Quad Tree for a set O of data points inside a n dimensional hyper cube μ is as follows: Let $\mu = [d_{1\mu} : d'_{1\mu}] \times [d_{2\mu} : d'_{2\mu}] \times \cdots \times [d_{n\mu} : d'_{n\mu}]$. If the number of data points in any bucket is less than threshold then the Quad Tree consists of a single leaf where the set O and the hypercube μ are stored. At each stage every bucket gets subdivided into 2^nsub buckets. Let us consider the division of buckets for $n = 2$. Let $\mu_{d_{1L}d_{2R}}, \mu_{d_{1R}d_{2R}}, \mu_{d_{1L}d_{2L}}, \mu_{d_{1R}d_{2L}}$ denote the four quadrants of μ (Fig. 1).

Let $d_{1mid} = (d_{1\mu} + d'_{1\mu})/2$ and $d_{2mid} = (d_{2\mu} + d'_{2\mu})/2$ and define

$$O_{d_{1R}d_{2R}} = \{o \in O : o_{d_1} > d_{1mid} \text{ and } o_{d_2} > d_{2mid}\},$$
$$O_{d_{1L}d_{2R}} = \{o \in O : o_{d_1} \leq d_{1mid} \text{ and } o_{d_2} > d_{2mid}\},$$
$$O_{d_{1L}d_{2L}} = \{o \in O : o_{d_1} \leq d_{1mid} \text{ and } o_{d_2} \leq d_{2mid}\},$$
$$O_{d_{1R}d_{2L}} = \{o \in O : o_{d_1} > d_{1mid} \text{ and } o_{d_2} \leq d_{2mid}\}.$$

Similarly for $n = 3$, eight sub buckets would be created namely,

$$\mu_{d_{1L}d_{2R}d_{3L}}, \mu_{d_{1L}d_{2R}d_{3R}}, \mu_{d_{1R}d_{2R}d_{3L}}, \mu_{d_{1R}d_{2R}d_{3R}},$$
$$\mu_{d_{1L}d_{2L}d_{3L}}, \mu_{d_{1L}d_{2L}d_{3R}}, \mu_{d_{1R}d_{2L}d_{3L}}, \mu_{d_{1R}d_{2L}d_{3R}}.$$

For n dimensional data set the sub buckets will be named as $\mu_{d_{1a}d_{2a}d_{3a}\ldots d_{na}}$, $a \in \{L, R\}$.

3.2 The Proposed Initialization Algorithm

First, some definitions of notations and parameters used in the initialization algorithm are provided.

Parameters and Definitions

MIN: user defined threshold for minimum number of data points in a sub bucket.

MAX: user defined threshold for maximum number of data points in a sub bucket.

δ: user specified distance for finding nearest neighbors.

White leaf bucket: a sub bucket having less than *MIN* percent of data points of the parent bucket.

Black leaf bucket: a sub bucket having more than *MAX* percent of data points of the parent bucket.

Gray bucket: a sub bucket which is neither white nor black.

$\Re_k$: neighborhood set of center c_k of a black leaf bucket.

C: set of cluster centers used for initializing K-Means algorithm.

Algorithm 1 gives the pseudocode for the initialization algorithm. In lines 1-8 of the algorithm, we divide an initial data space into buckets and continue until all buckets are either black or white leaf buckets as illustrated in Figs. 2a and 2b. In Fig. 2a the first division into four buckets is done. Out of these, three buckets are gray while one is white. In Fig. 2b the gray buckets are further subdivided, while the white one is left as such. At this stage, one of the sub buckets is labeled as a black leaf bucket.

Algorithm 1. The Initialization Algorithm
Input: Max%, Min%, Data set (O), δ
Output: Number of centers $|C|$ and the centers C

1. initialize the data space as a gray bucket;
2. while there are gray buckets
3. {
4. select a bucket;
5. divide it into 2^n sub buckets; // n is the dimension
6. label the sub buckets as white leaf bucket, black leaf bucket or gray bucket;
7. for every black leaf buckets calculate center ($c_{i,(1\leq i\leq m)}$); // m is the number of black leaf buckets;
8. }
9. $C = \Phi$;
10. label all centers $c_{i,(1\leq i\leq m)}$ as unmarked;
11. for $i = 1$ to m do $\Re_i = c_i$;
12. for each neighborhood $\Re_{i(1\leq i\leq m)}$
13. {
14. if there exist an unmarked center in $\Re_i$ then
15. {
16. while there is an unmarked center c_k in $\Re_i$ then
17. {
18. select c_k and label it as marked;
19. find δ-nearest unmarked neighbors of c_k and include them in $\Re_i$;
20. }

TABLE 1
Descriptive Statistics for All the Data Sets

	No. of Data	Attributes	Min	Median	Max
SYD1	150	X	3	103	177
		Y	0	56	177
SYD2	163	X	19	116	375
		Y	1	190	105

21. for all $c_k \in \Re_i$ calculate the mean m_i and call it the cluster center;
22. $C = C \cup \{m_i\}$;
23. }
24. }
25. return C and $|C|$;

Line 9 initializes the set of cluster centers to null set, and line 10 labels all black leaf bucket centers (as obtained in line 7) as unmarked. The initial neighborhood sets $\Re_i$ are set to include the black leaf bucket centers $c_{i,(1\leq i\leq m)}$(line 11). These neighborhoods sets will now be expanded to include the δ-nearest neighbors ofc_i. This is done in lines 14-20, where we select an unmarked center c_k, mark it, find its δ-nearest neighbors and include them in the set $\Re_i$. After one neighborhood set is exhausted, in lines 21-22, the mean of all centers in $\Re_i$ is calculated and included in the set of cluster centers C. This is done for all the neighborhood sets with unmarked centers. Note that some neighborhood sets may not get expanded at all if their initial centers (as initialized in line 11) get included into other neighborhood sets and eventually get marked. In other words, we group the centers of the black leaf buckets such that each group contains centers of adjacent black leaf buckets. Then, we calculate the mean of each group. The means are used as initial cluster centers for the K-Means algorithm. At the end of all iterations the algorithm returns the set C and the number of centers. The output of the Quad Tree algorithm presented in Algorithm 1 is the set of centers. We use such centers as the initial cluster centers for the original K-Means algorithm.

Complexity of the algorithm: The complexity of generating the black leaf buckets is $(b+1)vc$ where the depth of the tree is b, v is the number of data points, and c is a constant. Lines 12-24 generate the neighborhood sets of the m black leaf buckets and this takes $O(m^2)$ time. Hence, the complexity of our algorithm is $O((b+1)v+m^2)$. Assuming $m << v$, the complexity can be assumed to be $O((b+1)v)$.

Criteria for selecting the parameter δ: For selecting δ we consider the $l_{\min}$ and $l_{\max}$ as the minimum and maximum levels at which the black leaf buckets are created. Let p be the side length of the initial bucket and n is the dimension then $dia_{\max} = \sqrt{n}p/2^{l_{\min}}$ and $dia_{\min} = \sqrt{n}p/2^{l_{\max}}$. As a guiding rule it is suggested that δ be selected between $dia_{\min}$ and $dia_{\max}$.

4 Experimental Design

4.1 Data Sets

We conducted experiments on four real data sets to test our algorithm. These data sets are: AR3, AR4, AR5 available at [22] and Iris data set [7]. Of these, the first three data sets are related to software fault prediction. The synthetic two dimensional two class data sets (SYD1 and SYD2) have been taken to illustrate the initialization algorithm. For SYD1 we have generated three well-separated clusters with co variances -8.406, 9.483 and 22.585. The mean values of the three clusters for X and Y attributes are (158.166, 57.062), (102.640, 138.12), and (24.204, 11.136). For SYD2 we have generated four well-separated clusters with co variances -0.2025, -7.533, 6.365, and -6.385. The mean values of the four clusters for X and Y attributes are (24.77, 3.1), (195.40, 54.312), (92.60, 216.60), and (256.85, 200.10). Out of the total of 163 data in SYD2, 10 data have been introduced as noise. Descriptive statistics for all the synthetic data sets are given in Table 1.

TABLE 2
Confusion Matrix

		Predicted Labels	
		False (Non Faulty)	True (Faulty)
Actual Labels	False (Non Faulty)	(True Negative) A	(False Positive) B
	True (Faulty)	(False Negative) C	(True Positive) D

4.2 Metric Thresholds

In order to determine acceptable metrics thresholds, there are three methods described as follows [11]: 1) Experience and Hints from literature: The threshold values are specified according to the empirical researchers, previously introduced in the literature. 2) Tuning machine: This approach uses a repository of problematic items (faulty modules). Accordingly, there are chosen threshold values that maximize the number of correctly detected items. 3) Analysis of multiple versions: This method does not parameterize a strategy with several thresholds, but adds an important time viewpoint for each suspected entity. The dimensions and metrics we used in our experiments for AR# data sets are same as Catal et al. and are as follows: Lines of Code (LoC), Cyclomatic Complexity (CC), Unique Operator (UOp), Unique Operand (UOpnd), Total Operator (TOp), Total Operand (TOpnd). Threshold vector [LoC, CC, UOp, UOpnd, TOp, TOpnd] was chosen as [65, 10, 25, 40, 125, 70] [1]. For the Iris data set we have used all the four attributes.

4.3 Evaluation Parameters

A confusion matrix is formed as in Table 2. The Actual labels of data items are placed along the rows, while the predicted labels are placed along the columns. For example, a False Actual label implies that the module is not faulty. If a not faulty module (Actual label—False) is predicted as nonfaulty (Predicted Label—False) then we get the condition of cell A, which is True Negative, and if it is predicted as faulty (Predicted label—True) then we get the condition of cell B, which is False Positive. Similar definitions hold for False Negative and True Positive. The False positive rate is the percentage of not faulty modules labeled as fault prone by the model and the False negative rate is the percentage of faulty modules labeled as not fault prone and Error is the percentage of mislabeled modules. The following equations are used to calculate these FPR, FNR, Error, and Precision [1]

$$FPR = \frac{B}{A+B}, \tag{1}$$

$$FNR = \frac{C}{D+C}, \tag{2}$$

$$Error = \frac{B+C}{A+B+C+D}. \tag{3}$$

The above performance indicators should be minimized. A high value of FPR would lead to wasted testing effort while high FNR value means error prone modules will escape testing.

In this paper, for calculating the evaluation parameters, if any metric value of the centroid data point of a cluster was greater than the threshold, that cluster was labeled as faulty and otherwise it was labeled as nonfaulty. After this the predicted fault labels were compared with the actual fault labels. Note that the clusters can be labeled according to the majority of its members (by comparing with metrics thresholds) but this increases the complexity of the labeling procedure since all the modules in the cluster need to be examined.

TABLE 3
Gain Values for Various Data Sets

C#	AR3	AR4	AR5	Iris	SYD1	SYD2
1	0000.000	0.000.000	0.000.000	0.0000	0.0000	0.0000
2	8482.825	7629.863	**5024.245***	266.366	6935.020	15769.25
3	**9487.919***	**10204.855***	4254.536	**270.66***	**11079.62***	18207.11
4	9483.899	9418.109	4197.171	270.614	10988.015	**20228.27***
5	9402.292	9536.031	4066.295	268.483	10939.179	20132.95
6	8679.119	9466.211	3928.565	269.172	10865.840	19909.28
7	8577.743	9408.413	3450.422	267.164	10769.540	19852.80
8	8252.662	9300.438	3501.186	267.982	10672.308	19692.30
9	8213.447	9218.340	3633.281	264.876	10594.040	19628.38
10	8068.498	9268.624	3496.519	261.874	10570.032	19472.81
11	8152.776	9113.923	3367.876	260.001	10564.431	19174.45
12	7818.184	9028.644	3160.925	261.595	10361.688	19096.63

*C#: Number of clusters, * Maximum gain value.*

4.4 Gain

The optimal number of clusters is said to occur when the intercluster distance is maximized (or intercluster similarity is minimized) and the intracluster distance is minimized (or intracluster similarity is maximized). The clustering gain [21] attains a maximum value at the optimal number of clusters. The simplified formula for calculation of gain is as follows:

$Gain = \sum_{k=1}^{K} (v_k - 1)\|z_0 - z_0^k\|_2^2$, where K is the number of clusters, v_k is the number of data points present in kth cluster, z_0 is global centroid defined as $z_0 = \frac{1}{v}\sum_{i=1}^{v} o_i$ where v is the total number of data points, o is the data points and z_0^k denotes the centroid of the kth cluster, which is defined as $z_0^k = \frac{1}{v_k}\sum_{i=1}^{v_k} o_i^{(k)}$ where $o_i^{(k)}$ denotes data points belongs to kth cluster.

4.5 Experimental Setup and Results

Table 3 presents the gain values for all the data sets as obtained by the simple K-Means algorithm. Values have been taken for up to 12 clusters. For each cluster, six runs have been executed and the maximum gain value has been reported. Initialization has been done by random selection of the initial cluster centers. For the Quad Tree-based algorithm there are four input parameters: *MIN*, *MAX*, *O*, and δ. The value for *MIN* has been chosen as 5 percent, and for MAX it is 95 percent. In the QDK algorithm, for AR3, AR4, AR5, Iris, SYD1, and SYD2 the δ values are 40, 80, 40, 0.55, 70, and 120, respectively, and the number of cluster centers obtained was 3, 3, 2, 3, 3, and 4, respectively. Table 5 fifth column presents the gain values obtained by applying QDK algorithm on various data sets. To be able to compare our clustering quality with the K-Means algorithm, we adjusted the threshold parameter δ to obtain the same number of clusters (3 for AR3, 3 for AR4, 2 for AR5, 3 for Iris, 3 for SYD1, and 4 for SYD2) which gave maximum gain values for K-means algorithm. Table 4 presents the prediction error analysis for the QDK approach as compared to other approaches, namely, two stage approaches with simple K-Means with six attributes (KM), Catal et al. Two stage approach (CT) [1], Catal et al. Single stage approach (CS) [1], Naïve Bayes (NB) and Linear discriminant analysis (DA) (with ten fold cross validation setting). QDK, KM, CT, and CS approach, as well as NB and DA have considered six attributes from the mentioned data sets.

To compare the performance of QDK for initialization of K-Means, we have conducted experiments in which QDK and two other initialization techniques GM (Likas et al. Global K-Means algorithm [23], [24]) and DD (SAS 2004 [23], [25]) have been executed. The distance parameter d for the DD algorithm has been obtained by multiple runs to obtain the desired number of clusters. These distance values have been mentioned in Table 5. The

TABLE 4
Software Fault Prediction Error Analyses

Dataset	Parameters	QDK%	KM%	CT%	CS%	NB %	DA%
AR3	FPR	34.54	34.54	44.09	43.63	09.00	3.60
	FNR	25.00	25.00	25.00	25.00	25.00	75.0
	Error	33.33	33.33	41.67	41.27	11.10	12.60
AR4	FPR	4.59	31.03	34.83	35.63	05.70	3.40
	FNR	45.0	20.00	20.00	20.00	55.00	60.00
	Error	12.14	28.97	32.06	32.71	10.20	14.00
AR5	FPR	14.28	14.28	28.92	32.14	14.20	07.14
	FNR	12.50	12.50	12.50	12.50	12.50	37.5
	Error	13.88	13.88	25.28	27.70	13.88	13.80

parameters for evaluation are number of iterations (NOI) which counts the number of iterations of K-Means to arrive at the convergence criteria, Sum of squares error (SSE) [23], Gain and percent Error. The results are presented in Table 5. To check the applicability of density-based algorithms, we have implemented and applied the DBSCAN algorithm [6] on all the three data sets. The parameters *MinPts* (6) and *eps* (0.9) have been chosen heuristically by experiment. C++ programs ware developed to apply QDK, KM, GM, DD, and DBSCAN and to check the metrics thresholds. The NB and DA results have been obtained using DTREG (www.dtreg.com/DownloadDemo.htm) and WEKA (http://www.cs.waikato.ac.nz/ml/weka/). In this study, all the results were obtained using a desktop computer with 1 GB of RAM, Intel Pentium 4, 1.3 GHz CPU, Windows XP professional version 2002 Service Pack 1.

TABLE 5
Various Results of Different Initialization Techniques

Data	Techniques	N OI	SSE	Gain	Error
AR3	QDK (δ =40)	4	4110.337	9487.919	33.33
	KM*	6	4110.337	9487.919	33.33
	GM	592	4110.337	9487.919	33.33
	DD (d=100)	6	4110.337	9487.919	33.33
AR4	QDK (δ =80)	4	6596.423	8432.077	12.14
	KM*	8	5639.824	10204.85	28.97
	GM	1213	5639.824	10204.85	28.97
	DD (d=100)	7	5639.824	10204.85	23.36
AR5	QDK (δ =40)	5	2512.944	5024.245	13.88
	KM*	4	2512.944	5024.245	13.88
	GM	113	2512.944	5024.245	13.88
	DD (d=100)	3	2512.944	5024.245	13.88
Iris	QDK (δ =0.55)	10	97.3462	264.090	11.33
	KM*	11	97.3462	264.090	11.33
	GM	1726	97.3462	264.090	11.33
	DD (d=10)	13	97.3462	264.090	11.33
SYD 1	QDK (δ =70)	2	1513.813	110.79.62	03.11
	KM*	3	1513.813	110.79.62	03.11
	GM	793	1513.813	110.79.62	03.11
	DD (d=150)	4	1513.813	110.79.62	03.11
SYD 2	QDK (δ =120)	2	2158.433	20239.39	09.11
	KM*	4	2161.433	20228.27	10.56
	GM	1755	2161.433	20228.27	10.56
	DD (d=190)	3	2159.227	20230.11	10.44

* *Values taken from the best of six runs.*

5 Analysis of the Results

In Table 3 the maximum gain values obtained by multiple runs of K-Means algorithm for different cluster values are shown in bold face in each column. Table 5 fifth column shows the gain values obtained by the QDK algorithm for the same optimal number of clusters obtained by Table 3. These two values within each column are comparable. In fact gain values for QDK are nearly close or equal in all the cases except AR4 data set. This indicates that cluster quality obtained by QDK is comparable with K-Means algorithm. Table 4 presents the software fault prediction error analysis for three data sets namely AR3, AR4, and AR5. The values of FPR, FNR, and Error are presented for six techniques. The FPR values for QDK algorithm are better than CT and CS for AR3, AR4, and AR5 data sets and FNR values are same as CT and CS except in the case of AR4 data sets. The overall error rates are better than CT and CS for AR3, AR4 as well as AR5 data sets. Further, for AR4 and AR5 data sets, our approach is comparable to supervised learning approaches NB and DA. The overall error for AR4 data set is 12.14 for QDK and 10.20 for NB, while it 13.88 for both QDK and NB in AR5 data set. The overall error for AR4 data set is better (12.14) for QDK as compared to DA (14.00). The overall error for AR5 is 13.88 for QDK and 13.80 for DA which is very close. The percent Error of QDK is equal to that of KM for AR3 and AR5 data sets while it performs better in AR4 data set with respect to KM. Moreover, QDK performs better than CT and CS for all the data sets. For comparison sake we also labeled the clusters by majority labeling and the cluster labels matched exactly with the labels we had earlier assigned by mean. However, we did not proceed with this approach since it required inspecting all the data points in a cluster while our method required only one data which is the mean of the final clusters. Results of Table 5 show that the NOI for QDK is best for all the data sets except AR5 data set. The SSE and Gain values of QDK are equal or comparable for all data sets except AR4. The DBSCAN algorithm was applied upon the three data sets and the number of clusters obtained was 1, 2, and 2 for AR3, AR4, and AR5 data sets, respectively. It may be noted from Table 3 that the highest gain values for the three data sets were obtained at 3, 3, and 2 clusters. Moreover, DBSCAN treats some of the data as noise, for example, in AR3, AR4, and AR5 data sets 13, 1, 1 percent, data, respectively, were treated as noise data.

6 Conclusion

In this paper, we have evaluated the effectiveness of Quad Tree-based K-Means clustering algorithm in predicting faulty software modules as compared to the original K-Means algorithm. Quad Trees are applied for finding the initial cluster centers for K-Means algorithm. In case the user intends to form a desired (say$K$) number of clusters for K-Means algorithm, the Quad Tree-based algorithm can give K initial cluster centers to be used as input to the simple K-Means algorithm. This is facilitated by varying the value of the threshold parameter which is input to the Quad Tree algorithm. The overall error rates of software fault prediction approach by QDK algorithm are found comparable to other existing algorithms and are presented in Table 4. In fact, in the case of AR4 and AR5 data sets, the overall error rates of QDK are comparable with the supervised learning approaches NB and DA. The results of Table 5 show that the QDK algorithm works as an effective initialization algorithm. The number of iterations of K-Means algorithm is less in the case of QDK except for AR5, and the SSE as well as percent Error also give fairly acceptable values.

Acknowledgments

This research work has been partially funded by University Grant Commission [F.No.: 33-61/2007 (SR)] under financial grants for Major Research Project. The authors thank their postgraduate student, Mr. Akhilesh K. Yadav, for his assistance.

References

[1] C. Catal, U. Sevim, and B. Diri, "Clustering and Metrics Threshold Based Software Fault Prediction of Unlabeled Program Modules," *Proc. Sixth Int'l Conf. Information Technology: New Generations,* pp. 199-204, 2009.
[2] S. Zhong, T.M. Khoshgoftaar, and N. Seliya, "Unsupervised Learning for Expert-Based Software Quality Estimation," *Proc. IEEE Eighth Int'l Symp. High Assurance Systems Eng.,* pp. 149-155, 2004.
[3] S. Zhong, T.M. Khoshgoftaar, and N. Seliya, "Analyzing Software Measurement Data with Clustering Techniques," *IEEE Intelligent Systems,* vol. 19, no. 2, pp. 20-27, Mar./Apr. 2004.
[4] N. Seliya and T.M. Khoshgoftaar, "Software Quality Classification Modeling Using the PRINT Decision Algorithm," *Proc. IEEE 14th Int'l Conf. Tools with Artificial Intelligence,* pp. 365-374, 2002.
[5] J. Han and M. Kamber, *Data Mining Concepts and Techniques,* second ed, pp. 401-404. Morgan Kaufmann Publishers, 2007.
[6] M. Ester, H.P. Kriegel, J. Sander, and X.Xu., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD '96),* pp. 226-231, 1996.
[7] http://archive.ics.uci.edu/ml/datasets/Iris, 2012.
[8] P.S. Bishnu and V. Bhattacharjee, "A New Initialization Method for K-Means Algorithm Using Quad Tree," *Proc. Nat'l Conf. Methods and Models in Computing (NCM2C),* pp. 73-81, 2008.
[9] M. Laszlo and S. Mukherjee, "A Genetic Algorithm Using Hyper-Quadtrees for Low-Dimensional K-Means Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 4, pp. 533-543, Apr. 2006.
[10] D. Pelleg and A. Moore, "X-Means: Extending K-Means with Efficient Estimation of the Number of Cluster," *Proc. 17th Int'l Conf. Machine Learning,* pp. 727-734, 2000.
[11] R. Marinescu, "Detection Strategies: Metrics-Based Rules for Detecting Design Flaws," *Proc. 20th Int'l Conf. Software Maintenance,* pp. 350-359, 2004.
[12] N. Seliya, T.M. Khoshgoftaar, and S. Zhong, "Analyzing Software Quality with Limited Fault- Proneness Defect Data," *Proc. IEEE Ninth Int'l Symp. High-Assurance Systems Eng.,* pp. 89-98, 2005.
[13] K.E. Emam, S. Benlarbi, and N. Goel, "Comparing Case Based Reasoning Classifiers for Predicting High Risk Software Component," *J. Systems and Software,* vol. 55, no. 3, pp. 301-320, 2001.
[14] V. Bhattacherjee, P.K. Mohanti, and S. Kumar, "Complexity Metrics for Analogy Based Effort Estimation," *J. Theoretical and Applied Information Technology,* vol. 6, no. 1, pp. 001-008, 2009.
[15] S. Vicinanza, M.J. Prietulla, and T. Mukhopadhyay, "Case Based Reasoning in Software Effort Estimation," *Proc. 11th Int'l Conf. Information Systems.* pp. 149-158, 1990.
[16] V. Bhattacherjee and P.S. Bishnu, "Unsupervised Learning Approach to Fault Prediction in Software Module," *Proc. Nat'l Conf. Computing and Systems,* pp. 101-108, 2010.
[17] S. Wang and M.P. Armstrong, "A Quad Tree Approach to Domain Decomposition for Spatial Interpolation in Grid Computing Environment," *J. Parallel Computing: High Performance Computing with Geographical Data:* vol. 29, no. 10, pp. 1481-1504, 2003.
[18] M.D. Berg, O. Cheong, M. Kreveld, and M. Overmars, *Computational Geometry Algorithms and Applications,* third ed., pp. 309-315. Springer, 2008.
[19] R.A. Finkel and J.L. Bentley, "Quad Trees: A Data Structure for Retrieval on Composite Key," *Acta information,* vol. 4, no. 1, pp. 1-9, 1974.
[20] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the Number of Clusters in a Dataset via the Gap Statistic," *J. Statistical Soc.,* vol. 63, no. 2, pp. 411-423, 2001.
[21] Y. Jung, H. Park, and D.Z. Du, "A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering," *J. Global Optimization,* vol. 25, pp. 91-111, 2003.
[22] http://promisedata.org/, 2012.
[23] D. Steinley and M.J. Brusco, "Initializing K-Means Batch Clustering: A Critical Evaluation of Several Techniques," *J. Classification,* vol. 24, pp. 99-121, 2007.
[24] A. Likas, N. Vlassis, and J. Verbeek, "The Global K-means Clustering Algorithm," *Pattern Recognition,* vol. 36, pp. 451-461, 2003.
[25] SAS, "The FASTCLUS Procedure," in SAS/STAT 9.1 User's Guide vol. 2, Cary, NC, SAS Inst., Inc., 2004.
[26] P.S. Bishnu and V. Bhattacherjee, "Outlier Detection Technique Using Quad Tree," *Proc Int'l Conf. Computer Comm. Control and Information Technology,* pp. 143-148, Feb. 2009.
[27] P.S. Bishnu and V. Bhattacherjee, "Application of K-Medoids with kd-Tree for Software Fault Prediction," *ACM Software Eng. Notes,* vol. 36, pp. 1-6, Mar. 2011.
[28] V. Bhattacherjee and P.S. Bishnu, "Software Fault Prediction Using K-Medoids Algorithm," *Proc. Int'l Conf. Productivity, Quality, Reliability, Optimization and Modeling (ICPQROM '11),* p. 191, Feb. 2011.