International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)

http://www.ijcsjournal.com Volume 3, Issue 1, No 4, 2015. ISSN: 2348-6600
Reference ID: IJCS-089 PAGE NO: 510-514.

Software Process Quality Assessment in Medium Sized Software Organizations

A.SARANYA[#1], Dr.S.KANNAN[#2]
[#1]Research Scholar in Computer Science, [#2]Associate Professor
[#]Department of computer Application
Madurai Kamaraj University
Madurai, Tamil Nadu, India
[1]asaranyaalagar@gmail.com
[2]skannanmku@gmail.com

Abstract— The spurt in the number of medium-sized software organizations has thrown open a number of challenges unique to this segment. These organizations are faced with the twin issues of reducing the costs and achieving quality products at the same time. Most Software processes are typically applicable only for large organizations that can afford the incurred cost. In this context, software process improvement in such organizations is of vital importance. This paper proposes the use of genetic algorithm together with the clustering data mining technique to estimate the quality of processes used by a medium sized organization. Such estimation of process quality would definitely aid the organization in identifying the faulty processes and thereby contribute to process improvement in future projects.

1. Introduction

Medium sized software organizations face a host of novel challenges different from large ones. Such organizations cannot afford to incur additional costs incurred by software process improvement initiatives widely available in the literature. Advancements in the computer science discipline, notably, genetic algorithms and data mining can be of immense utility in addressing various SE problems. The present paper is an attempt to exploit the power of these disciplines in addressing the software process improvement challenge faced by medium sized software organizations.

Clustering is a division of data into groups of similar objects. Each group called a cluster consists of objects that are similar between themselves and dissimilar to objects of other groups [1]. These clusters correspond to hidden patterns, and the search for clusters is termed “unsupervised learning”. One of the most popular clustering algorithms is the k-means clustering algorithm.

1.1 k-means Clustering

k-means clustering algorithm follows a simple way to classify a given data set thorogh a certain number of clusters fixed apriori. The algorithm starts by defining k-centroids, one for each cluster. The better choice to place the centroids is to place them as far as possible from each other. The algorithm then proceeds to take each point in the data set and associate it with the nearest centroid. When all points are done this way, the first iteration is completed and an early groupage is done. Now the algorithm recalculates k new centroids. After this a new binding has to be done between the same set of data points and the new centroids. The k-centroids change step by step until no more changes are done. The algorithm aims at minimizing an objective function which is the squared error function. The objective function

$$j = \sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i^{(j)} - c_j\right\|^2$$ where $\left\|x_i^{(j)} - c_j\right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the n data points from their respective cluster centres.

1.2 Genetic k-means clustering

Krishna and Murty propose a novel hybrid genetic algorithm that finds a globally optimal partition of a given data into specified number of clusters [2]. They attempt to hybridize GA with the k-means algorithm. The important aspects of the proposed GKA are listed below:

- Coding – W is encoded into a string s_W by considering a chromosome of length n and allowing each allele to take values {1,2, …, K}. Each allele represents a pattern and the allele value indicates the cluster number to which the pattern belongs. This is called string-of-group-numbers encoding.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 4, 2015. ISSN: 2348-6600
Reference ID: IJCS-089 PAGE NO: 510-514.

- Initialization – as with most GA's the initial population is obtained by initializing each allele in the population to a random number selected from the set {1,2, …, K}.

- Selection – a chromosome is selected from the previous population according to the distribution: $P(s_i) = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)}$ where $F(s_i)$ represents the fitness value.

- Fitness function – in order to minimize S(W) – the Total Within Cluster Variation, Krishna and Murty resort to the σ-truncation mechanism. They define $f(s_W) = -S(W), g(s_W) = f(s_W) - (\chi - c.\sigma)$ where χ and σ denote the average value and standard deviation of $f(s_W)$ in the current population and c is a constant between 1 and 3. Fitness is calculated as: $F(s_W) = \begin{cases} g(s_W), if\ g(s_W) \geq 0 \\ 0, \quad otherwise \end{cases}$

- Mutation – the main issue concerning the mutation operator is that the probability of changing an allele value must be more if the corresponding cluster centre is closer to the data point. In order to define the mutation operator, Krishna and Murty define $d_j = d(x_i, c_j)$ as the Euclidean distance between x_i and c_j. the replacement value for an allele is chosen according to the distribution: $p_j = \Pr\{s_W(i) = j\} = \frac{c_m d_{\max} - d_j}{\sum_{i=1}^{K} (c_m d_{\max} - d_i)}$ where c_m is a constant usually >= 1 and $d_{\max} = \max\{d_j\}$. Krishna and Murty also ensure that empty clusters are not formed by proposing to change the allele only if the distance between the data point x_i and the corresponding cluster centre $c_{sW}(i)$ is greater than zero.

- The k-means operator – as the GA with just the above steps may take more time to converge, Krishna and Murty propose the k-means operator described as follows: let s_W be the string. The k-means operator (KMO) when applied to s_W produces s_W^* according following the steps listed below: a) For the given matrix W, cluster centers are calculated b) each data point is reassigned to the nearest cluster with cluster centre.

Krishna and Murty prove that their proposed GKA performs better than the ordinary k-means algorithm, Evolutionary Strategies and Evolutionary Programming. They also claim that the method is computationally less expensive.

2. Related Work

2.1 Automatic Clustering of Software Systems using a Genetic Algorithm

Doval, Mancoridis and Mitchell propose a novel method for automatic clustering of Software Systems [3]. Designers use directed graphs to represent the inter-relationships between modules. Such graphs are called Module Dependency Graphs (MDGs). Doval et.al address the problem of partitioning the MDG to find "good" MDG Partitions.

Module Dependency Graphs are often used to represent the structure of complex systems. Modules are represented as nodes and their relationships are represented as edges. The complexity of the MDG of a modest sized system can itself be quite daunting and therefore, finding good partitions of the MDG where each partition can be relatively simpler compared to the entore system can be extremely useful. The huge dimensionality imposed by MDG's necessitates the use of a Genetic Algorithm to perform clustering.

2.2 Software Quality Estimation with Clustering

Zhong, Khoshgoftaar and Seliya claim that Software Quality Estimation using supervised techniques like classification works fine only if past data from similar software projects are available [4]. They attempt to use cluster analysis with expert input for predicting the fault-proneness of software modules.

The basic idea behind the work of Zhong et.al. is as follows: In the absence of software proneness labels, which might result from an organization dealing with a particular type of project it has not dealt with before, Unsupervised learning method like cluster analysis represent a viable option. Software modules are grouped into clusters based on the

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 4, 2015. ISSN: 2348-6600
Reference ID: IJCS-089 PAGE NO: 510-514.

values of various metrics. Modules that are fault-prone will have similar values of metrics and thus be grouped in one cluster. Modules that are not fault-prone will constitute another cluster. After the cluster analysis is complete, an expert can inspect each cluster and label it as "fault-prone" or "not-fault-prone". Exercising this option saves a lot of effort for the expert who would otherwise have to analyze every module to determine it's fault-proneness.

For a module under study, the various metrics associated with it can provide indications of various aspects of it. For example, the McCabe's Cyclomatic Complexity provides insight into the complexity associated with the module. This complexity may have implications for understandability and maintainability. For module clustering, the values of these metrics form the basic input based upon which it is placed in a particular cluster.

3. Proposed Methodology

The basic methodology is very similar to the one adopted by Zhong et.al. The main objective was to determine whether the genetic k-means algorithm can be applied successfully to estimate the quality of processes employed by medium sized organizations.. Zhong et.al. also study the Neural Gas clustering algorithm but this paper does not consider that. A medium sized local software organization involved in providing software solution to its vendors was subject to the study. The organization is responsible for the development and maintenance of web sites for it's clients. Data pertaining to the processes applied for engineering around 50 web sites developed by the organization are studied. The Software Measurement data pertaining to the processes adopted for all the 50 applications were collected

A set of 7 measures for each of the processes followed was collected. These include: Number of errors found in the requirements document, Number of errors uncovered at the design phase, Number of errors detected at coding phase, whether or not peer reviews were conducted during coding, Number of errors uncovered at testing phase, engagement of an independent testing team different from the original development team and adoption of configuration management practices.

The 7 metrics taken into account for the study do not in any way fully characterize a software process. Many metrics are available and some of them may be even more accurate predictors of the attribute they are attempting to measure. The choice of these metrics was primarily based on the interactions with the expert involved in labeling. The expert was the senior project manager with 10 years of experience in web site development, maintenance and management.

Both the k-means and the genetic k-means clustering algorithms were implemented in Java under the Windows XP Operating System. The number of clusters to be generated heavily depends on the availability of resources. For the experiment, the number of clusters was fixed at 5. The choice of the number of clusters is a trade-off between the effort required by the expert for labeling the clusters and obtaining a fine representation of the software data. The more the number of clusters, more is the effort required by the expert and finer the granularity.

For labeling, the expert is provided with the global mean, minimum, maximum, median, 75^{th} percentile, 80^{th} percentile, 85^{th} percentile and 90^{th} percentile of each metric for each cluster as well as the size of each cluster. Based on these statistics, the expert labeled each cluster as "quality" or "not quality". This is the same methodology followed by Zhong et.al.

For the comparative analysis, the following measurements are used. These are essentially the same ones used by Zhong et.al.

- Mean Squared Error (MSE)
- Cluster Purity – percentage of the most dominated category (fault-prone or not-fault-prone) in the cluster
- Average Purity – Mean purity of all the clusters

The expert's labeling decision is evaluated using the following criteria:

- Over-all classification error
- False Positive rate (FPR) – percentage of not quality processes mislabeled as quality
- False-Negative Rate (FNR) – percentage of quality modules mislabeled as not-quality.

For the experiment, a process was considered "not quality" if the resulting website had more than 10 issues reported after delivery. This was used only to assess the labeling performance of the expert and not for clustering.

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 4, 2015. ISSN: 2348-6600
Reference ID: IJCS-089 PAGE NO: 510-514.

3.1 Results and Discussion

The results of the experiment are tabulated below:

Table 1 - Clustering Quality Results

Measure	k-means	Genetic k-means
Mean Squared Error	1391.81	1273.35
Average Purity	0.743	0.832

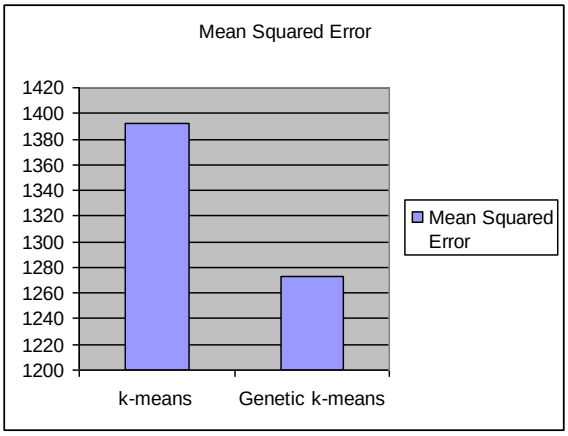

Figure 1 – Mean Squared Error when applying k-means and Genetic k-means

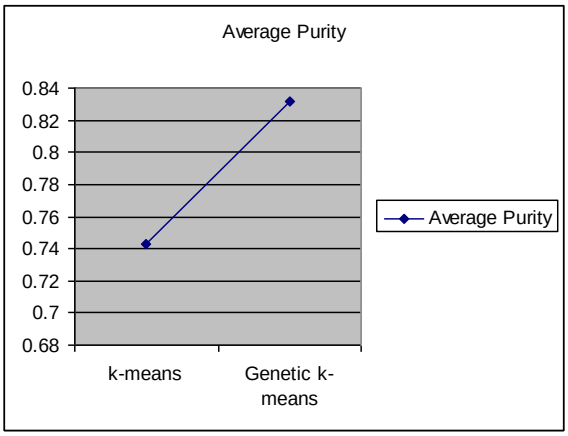

Figure 2 – Average Cluster Purity when applying k-means and Genetic k-means

Table 2 – Expert Labeling Performance

	False Positive (%)	False Negative (%)	Overall (%)
k-means	15.8%	15.2%	21.6%
Genetic k-means	13.9%	13.7%	20.9%

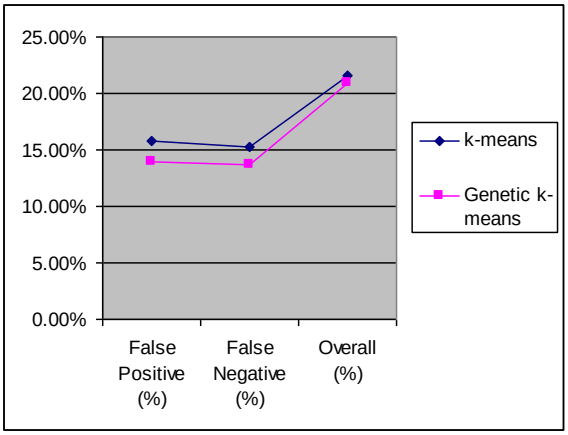

Figure 3 – Expert labeling performance

A closer analysis of the results reveals certain interesting properties of the Genetic k-means algorithm. First, considering the clustering quality, it is observed that Genetic k-means tends to produce clusters with a lower mean squared error value compared to k-means. Genetic k-means produces clusters with more average purity

When analyzing the expert labeling decision, the first thing to be noted is that the feedback from the expert indicated that clusters generated by genetic k-means were more easier to label that those generated by k-means. This tends to be because of the ability of the genetic k-means algorithm to produce more coherent clusters. Narrowing the analysis to false-negative misclassification rates, as the cost of such misclassifications is significantly higher than that of false-positive misclassifications, the genetic k-means is found to perform better.

3.8 Conclusion

The potential of the genetic k-means algorithm to cluster software processes based on their measures was studied. It was found that the genetic k-means algorithm performs significantly better than k-means. Given the paucity of resources in most medium sized organizations, such ability to predict the quality of processes is vital, so that the organization can avoid repeating the faulty processes and whenever possible address the issues in them. Clustering has the added advantage that it can perform even in the absence of data of similar past software projects. In this regard, the results of the conducted empirical study demonstrate that the genetic k-means can prove to be more useful and accurate than k-means clustering.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)

http://www.ijcsjournal.com Volume 3, Issue 1, No 4, 2015. ISSN: 2348-6600
Reference ID: IJCS-089 PAGE NO: 510-514.

References

[1] Berkhin, Pavel, Survey of Clustering Data Mining Techniques, Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.895

[2] Krishna, K, Murty M,N, "genetic k-means algorithm", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol 29, Issue 3, 199, pp 433-439

[3] D. Doval, S. Mancoridis, B. S. Mitchell, Automatic Clustering of Software Systems using a Genetic Algorithm, Retrieved from: https://www.cs.drexel.edu/~spiros/papers/step99.pdf

[4] Zhong, Shi, Khoshgoftaar. Taghi M, Seliya, Naeem, "Analyzing Software Measurement Data with Clustering Techniques", IEEE Intelligent Systems, IEEE Computer Society, 2004