International Journal of Computer Applications (0975 – 8887)
Volume 8– No.13, October 2010

Evaluation of Expectation Maximization based Clustering Approach for Reusability Prediction of Function based Software Systems

Dr Himani Goel
Department of Computer science & Engg
B.M.S.C.E Muktsar, Punjab (India)

Gurbhej Singh
Department of Computer science & Engg
B.M.S.C.E Muktsar, Punjab (India)

ABSTARCT

In this study Expectation Maximization based Clustering approach is evaluated for Reusability Prediction of Function based Software systems. Here, the metric based approach is used for prediction. The function oriented dataset considered have the output attribute as Reusability value. The Reusability in the dataset is expressed in terms of six numeric labels i.e. 1, 2, 3, 4, 5 and 6. The label 1 represents Nil and the label 6 represents the Excellent Reusability Label. A framework of metrics are used to target those the essential attributes of function oriented features towards measuring the reusability of software modules, so it tried to analyze, refine and use following metrics to explore different structural dimensions of Function oriented components: Cyclometric Complexity Using Mc Cabe's Measure, Halstead Software Science Indicator, Regularity Metric, Reuse-Frequency Metric and Coupling Metric. The input attributes are expressed in the three linguistic labels i.e. 1, 2, and 3. The label 1 corresponds to the Low value, label 2 corresponds to the Medium value and label 3 corresponds to the high value.Five Input metrics are used as Input and clusters are formed using EM. EM assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters.Thereafter 10 fold cross validation performance of the system is recorded. The results are expressed in Precision, Recall and Accuracy values. Precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). Hence, Precision can be seen as a measure of exactness or fidelity, whereas Recall is a measure of completeness. Accuracy is the percentage of the predicted values that match with the expected values of the reusability for the given data. As deduced from the results it is clear that Precision and Recall values of the sixth level reusability class is the maximum, it means the system is able to detect the "Excellent" components precisely. Similarly, Precision and Recall values of the fourth level reusability class is the second best, it means the system is able to detect the "Good" components with good Precision. The proposed technique is showing Accuracy value approximately equal to 60%, so it is satisfactory enough to use the Expectation maximization based clustering technique for the prediction of the function based reusable modules from the existing reservoir of software components. The proposed approach is applied on the C based software modules/components and it can further be extended to the Artificial Intelligence (AI) based software components e.g. Prolog Language based software components. It can also be tried to calculate the fault-tolerance of the software components with help of the proposed metric framework.

Keywords

Fault Prediction, Cyclometric Complexity, volume, Regularity Metric, Reuse-Frequency Metric, Coupling Metric, Precision Recall ,Accuracy, confusion matrix, expectation maximisation.

1 INRODUCTION

Software reuse (Frakes, W.B., 2005) is the process of implementing or updating software systems using existing software assets. Software assets or components include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites. Anything that is produced from a software development effort can potentially be reused.The reusability is the quality of a piece of software, that enables it to be used again, be it partial, modified or complete. Software professionals have recognized reuse as a powerful means to potentially overcome the situation called as software crisis. Software reuse not only improves productivity but also has a positive impact on the quality and maintainability of software products.There are two approaches for reuse of code: develop the code from scratch or identify and extract the reusable code from already developed code. For the organization that has experience in developing software, but has not yet used the software reuse concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir. The cost of developing the software from scratch can be saved by identifying and extracting the reusable components from already developed software systems or legacy systems .The contribution of metrics to the overall objective of the software quality is very well understood and recognized. But how these metrics collectively determine reusability of a software component is still at its naïve stage.
There are two forms of reuse and they are as:

- Horizontal Reuse.
- Vertical Reuse.

Horizontal reuse refers to software components used across a wide variety of applications. In terms of code assets, this includes the typically envisioned library of components, such as a linked list class, string manipulation routines, or graphical user interface (GUI) functions. Horizontal reuse can also refer to the use of a commercial off-the-shelf (COTS) or third-party application within a larger system, such as an e-mail package or a word processing program. A variety of software libraries and repositories containing this type of code and documentation exist today at various locations on the Internet.

Vertical reuse, significantly untapped by the software community at large, but potentially very useful, has far reaching implications for current and future software development efforts.

1.1 Reuse Process

The process of reuse consists of four major activities:

- manage the reuse infrastructure (MRI).
- produce reusable assets (PRA).
- broker reusable assets (BRA) and
- consume reusable assets (CRA).

Producers are those who create reusable assets with the specific goal of reusability.

Function of Manage the Reuse Infrastructure (MRI) is to establish the reuse rules, roles, and goals in the infrastructure to support reuse. The Produce Reusable Assets (PRA) activities develop, generate, or reengineer assets with the specific goal of reusability. PRA includes domain analysis and domain engineering. The Broker Reusable Assets (BRA) activity aids the reuse effort by qualifying or certifying, configuring, maintaining, promoting and brokering reusable assets. The Consume Reusable Assets (CRA) activity occurs when systems are produced using reusable assets (Poulin, J. S., 1997).

Followings are the steps for implementing Reuse Process:

- Assess organizational readiness: Understand the people, process, product, technology, asset, economic, metric and management facets of the organization and how reuse will impact each of these aspects.
- Identify and collect metrics: While this activity is done throughout the reuse effort as necessary, collecting metrics early will enable us to benchmark the organization and show the impact when reuse is implemented.
- Identify domains in the organization: Enumerate a list of domains that are in common within the organization.
- Analyze the domain: An informal domain analysis may be conducted for the chosen domain. This analysis includes determining features common to systems in the domain and assessing the range of variability.
- Examine the existing organizational structure: Consider establishing an independent producer group. This would dedicate resources to ensure that the necessary assets are created, managed and supported.
- Create and manage reusable assets: Make, buy or re-engineer existing assets for users. Bring these assets under a source control and configuration management system.
- Utilize tools, technology, and standards: Examine whether to create or use existing tools, technology and standards for your reuse program.
- Conduct reviews and walkthroughs to reinforce reuse: Throughout the product development life cycles, perform reviews to ensure adherence to reusability objectives.

1.2 Introduction to Clustering Techniques

As a broad subfield of Fault Prediction, clustering is concerned with the design and development of algorithms and techniques that allow division of data in to different groups.

Clustering means to assign a set of observations in to different groups (known as clusters), so that the observations are same in some sense. At a general level, there are two types of clustering: distance based and conceptual clustering. Distance based clustering divides the data in to subsets on the basis of distance. Conceptual clustering, cluster the data on the basis of the similar concept the data will have.

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. It is the ordinary distance between two points that one would measure with a ruler, which can be proven by repeated application of the Pythagorean theorem. The major focus of clustering research is to extract information from data automatically, by computational and statistical methods. Hence, clustering is closely related to data mining and statistics.

Many clustering methods aim at finding a single partition of the collection of items into clusters. However, obtaining a hierarchy of clusters can provide more flexibility and other methods rather focus on this. A partition of the data can be obtained from a hierarchy by cutting the tree of clusters at some level. Most clustering methods were developed for numerical data, but some can deal with categorical data or with both numerical and categorical data.

The degree of membership of a data item to a cluster is either in [0, 1] if the clusters are fuzzy or in {0, 1} if the clusters are crisp. For fuzzy clusters, data items can belong to some degree to several clusters that don't have hierarchical relations with each other. This distinction between fuzzy and crisp can concern both the clustering mechanisms and their results. Crisp clusters can always be obtained from fuzzy clusters. Clusters can be seen either as distant compact sets or as dense sets separated by low density regions. Unlike density, compactness usually has strong implications on the shape of the clusters, so methods that focus on compactness should be distinguished from methods that focus on the density. Clustering denotes changes in a system that enables a system to do the same task more efficiently the next time. Clustering is a method of unsupervised learning, in which one seeks to determine how the data are organized.

Clustering algorithms can be:

1.2.1 Hierarchical:

A hierarchical algorithm creates a hierarchy of clusters which may be represented in a tree structure called a dendrogram. The root of the tree consists of a single cluster containing all observations, and the leaves correspond to individual observations. In hierarchical clustering algorithm, a valid metric may be used as a measure of similarity between pairs of observations. Algorithms for hierarchical clustering are generally either agglomerative, in which one starts at the leaves and successively merges clusters together; or divisive, in which one starts at the root and recursively splits the clusters.

1.2.2 Partitional:

Partitional algorithms typically determine all clusters at once. These algorithms divide data in to independent clusters on the basis of distance measures. A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.

K-Means is an unsupervised clustering technique used to classify data in to K clusters. It is partitional clustering approach, each cluster is associated with a centroid (center point), each point is assigned to the cluster with the closest centroid, Number of clusters, K, must be specified.

Fuzzy C-Means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. It processes n vectors in p-space as data input, and uses them, in conjunction with first order necessary conditions for minimizing the FCM objective functional, to obtain estimates for two sets of unknowns. FCM clustering is used to build fuzzy rule bases for fuzzy systems design; and there are numerous applications of FCM in virtually every major application area of clustering

1.2.3: Spectral

This clustering techniques make use of the spectrum of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions.The main requirements that a clustering algorithm should satisfy are scalability; dealing with different types of attributes; discovering clusters with arbitrary shape; minimal requirements for domain knowledge to determine input parameters; ability to deal with noise and outliers; insensitivity to order of input records; high dimensionality; interpretability and usability.

2 LITERATURE SURVEY

(Boetticher, G., et.al, 1993) assimilated knowledge about object oriented concepts, analysis and design. Explanation of various object-oriented metrics was also given such as class oriented metrics (CK metric), metrics for source code, testing, analysis model and design model. Advantages and disadvantages of each object-oriented metric was explained. Information regarding theoretical background of the reusability of software and Object oriented metrics for measuring size, complexity are also given in this reference.

(Boetticher, et.al, 1993) discussed various approaches for measuring software reusability, to build reusable components and to identify useful modules in existing programs. Taxonomy of reusability metrics was given which provide the attributes of reusable software. Two main methods are there to measure the reusability. One is Empirical methods stress objective, numerical, and repeatable metrics, such as those obtained by observing the module complexity or size. Other is Qualitative methods included (or even emphasized) subjective criteria, such as how well a module complies with a set of style, certification, quality guidelines, or simply agrees with the opinions of "experts." These are further divided into two categories module oriented and component oriented.

(Kartalopoulos, S. V. 1996) discussed REBOOT (reusability based on object oriented technique) that develop a taxonomy of reusability attributes. It provided reusability factors, a list of criteria for factor and a list of metrics for each criteria. Various object oriented concepts were defined in this paper, which are useful for finding the reusability. Information about software reuse, types of software reuse, requirements for building software reuse and management issues toward reusable software were also given. This also shows the advantages of this technique over other methods.

Prieto-Diaz and Freeman encouraged white-box reuse and identified five program attributes for evaluating reusability . The attributes used are:

- Program Size
- Program Structure
- Program Documentation
- Programming Language
- Reuse Experience

(Richard W. Selby, 2005) discussed that CK metric suit is able to target all the essential attributes of OO-based software. (Parvinder, 2005) used Tuned and Refined values of the following Metric suit for the reusability data Modeling:

- ➢ Weighted methods per class (WMC)
- ➢ Depth of inheritance tree (DIT)
- ➢ Number of Children (NOC)
- ➢ Coupling Between Object Classes (CBO)

Lack of Cohesion in Methods (LCOM)

3 PRESENT WORK

3.1 Problem Formulation

The aim of Metrics is to predict the quality of the software products. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc. The requirement today is to relate the reusability attributes with the metrics and to find how these metrics collectively determine the reusability of the software component. To achieve both the quality and productivity objectives it is always recommended to go for the software reuse that not only saves the time taken to develop the product from scratch but also delivers the almost error free code, as the code is already tested many times during its earlier reuse.

A great deal of research over the past several years has been devoted to the development of methodologies to create reusable software components and component libraries, where there is an additional cost involved to create a reusable component from scratch. That additional cost could be avoided by identifying and extracting reusable components from the already developed large inventory of existing systems. But the issue of how to identify good reusable components from existing systems has remained relatively unexplored. Our approach, for identification and evaluation of reusable software, is based on software models and metrics. As the exact relationship between the attributes of the reusability is difficult to establish so a Neural Network approach could serve as an economical, automatic tool to generate reusability ranking of software by formulating the relationship based on its training. When one designs with Neural Networks alone, the network is a black box that needs to be defined; this is a highly compute-intensive process. One must develop a good sense, after extensive experimentation and practice, of the complexity of the network and the learning algorithm to be used. With the objective of taking advantage of the features of the neural networks, in this study Neural Network based approach is used to economically determining reusability of software components in existing systems as well as the reusable components that are in the design phase. Inputs to Neural system, are provided in form of McCabe’s Cyclometric Complexity Measure for Complexity measurement, Regularity Metric, Halstead Software Science Indicator for Volume indication, Reuse Frequency metric and Coupling Metric values of the software component and output is be obtained in terms of reusability.

3.2 Methodology

Reusability evaluation System for function Based Software Components can be framed using following steps:

I) Selection and refinement of metrics targeting the quality of function based software system and perform parsing of the software system to generate the Meta information related to

that Software. The metric of the Parvinder et. al 2006 is used and the metrics are as under:
The proposed five metrics for function Oriented Paradigm are as follows:

The proposed metrics for Function Oriented Paradigm are as follows:

3.2.1 Cyclometric Complexity Using Mc Cabe's Measure

According to Mc Cabe, the value of Cyclometric Complexity (CC) can be obtained using the following equation:

$$CC = Number of\ predicate nodes + 1 \qquad (1)$$

Where predicate nodes are the nodes of the directed graph, made for the component, where the decisions are made.Hence, the value of CC of a software component should be in between upper and lower bounds as a contribution towards reusabilityIf CC is high with high regularity of implementation then there exists high functional usefulness.

3.2.2 Halstead Software Science Indicator

According to this metric volume of the source code of the software component is expressed in the following equation:

$$Volume = N1 + N2 \log 2(\eta 1 + \eta 2) \qquad (2)$$

Where, η1 is the number of distinct operators that appear in the program, η2 is number of distinct operands that appear in the program, N1 is the total number of operator occurrences and N2 is the total number of operand occurrences.

The high volume means that software component needs more maintenance cost, correctness cost and modification cost. On the other hand, less volume increases the extraction cost, identification cost from the repository and packaging cost of the component. So the volume of the reusable component should be in between the two extremes.

3.2.3 Regularity Metric

The notion behind Regularity is to predict length based on some regularity assumptions. As actual length (N) is sum of N1 and N2. The estimated length is shown in the following equation:

$$Estimated\ Lenght = N' = \eta 1 \log 2\, \eta 1 + \eta 2 \log 2\, \eta 2 \qquad (3)$$

The closeness of the estimate is a measure of the Regularity of Component coding is calculated as:

$$\text{Re}\, gularity = 1 - \{(N - N') / N] = N' / N \qquad (4)$$

The above derivation indicates that Regularity is the ratio of estimated length to the actual length. High value of Regularity indicates the high readability, low modification cost and non-redundancy of the component implementation [24].

Hence, there should be some minimum level of Regularity of the component to indicate the reusability of that component.

3.2.4 Reuse-Frequency Metric

Reuse frequency is calculated by comparing number of static calls addressed to a component with number of calls addressed to the component whose reusability is to be measured. Let N user defined components be X_1, X_2 … X_N in the system, where S_1, S_2 … S_M are the standard environment components e.g. printf in C language, then Reuse-Frequency is calculated as:

$$Reuse - Frequency = \frac{\eta(C)}{\frac{1}{M}\sum_{i=0}^{M} \eta(S_i)} \qquad (5)$$

Equation (5) shows that the Reuse-Frequency is the measure of function usefulness of a component. Hence there should be some minimum value of Reuse- Frequency to make software component really reusable [24].

3.2.5 Coupling Metric

Functions/methods that are loosely bound tend to be easier to remove and use in other contexts than those that depend heavily on other functions or non-local data. Different types of coupling effects reusability to different extent.

Data Coupling:

Stamp Coupling:

Control Coupling:.

Common Coupling:

Data Coupling is lightest weight coupling, whereas Content Coupling is the heaviest one.

Let

a_i be the number of functions called and Data Coupled with function "i"

b_i be the number of functions called and Stamp Coupled with function "i"

c_i be the number of functions called by function "i" and Control Coupled with function "i"

d_i be the number of functions Common Coupled with function "i"

$$f(x, a, c) = \frac{1}{1 + e^{-a(w_1 a_i + w_2 b_i + w_3 c_i + w_4 d_i - c)}} \qquad (6)$$

Where a = 10, c = 0.5 and w_i for i = 1, 2, 3, 4 is the weights of the respective the coupling types.

As coupling increases, there is decrease in understandability and maintainability, so there should be some maximum value of the coupling.
II) Calculate the metric values of the sampled software components.
III) Implement the EM (expectation maximisation) Clustering based prediction system in Matlab environment.
EM assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters. EM can decide how many clusters to create by cross validation, or you may specify apriori how many clusters to generate.
The cross validation performed to determine the number of clusters is done in the following steps:
1. the number of clusters is set to 1

2. the training set is split randomly into 10 folds.
3. EM is performed 10 times using the 10 folds the usual CV way.
4. the loglikelihood is averaged over all 10 results.
5. if loglikelihood has increased the number of clusters is increased by 1 and the program continues at step 2.

The number of folds is fixed to 10, as long as the number of instances in the training set is not smaller 10. If this is the case the number of folds is set equal to the number of instances.
Deduce the results on the 10 fold cross validation accuracy, precision and recall values.

In case of the two-cluster based problem, the confusion matrix has four categories: True positives (TP) are modules correctly classified as faulty modules. False positives (FP) refer to fault-free modules incorrectly labeled as faulty modules. True negatives (TN) correspond to fault-free modules correctly classified as such. Finally, false negatives (FN) refer to faulty modules incorrectly classified as fault-free modules as shown in table 3.1.

Table 3.1. Confusion Matrix of Prediction Outcomes.

	Real Data Value	
Predicted Value	**Fault**	**No fault**
Fault	TP	FP
No Fault	FN	TN

With help of the confusion matrix values the precision and recall values are calculated described below:

- Precision

The Precision is the proportion of the examples which truly have class x among all those which were classified as class x.

Precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). The equation is:

Precision = TP / (TP + FP) (1)

- Recall

Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been) [8]. The recall can be calculated as follows:

Recall = TP / (TP + FN) (2)

- Accuracy

The percentage of the predicted values that match with the expected values of the reusability for the given data.

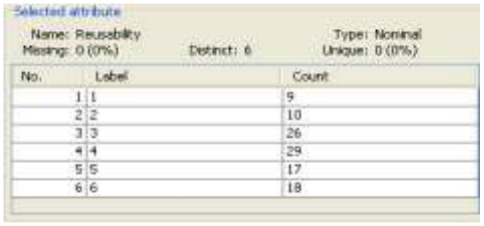
The best system is that having the high Accuracy, High Precision and High Recall value.

4 RESULTS AND DISCUSSION

The proposed Neural based methodology is implemented in MATLAB 7.4. MATLAB (Matrix Laboratory) environment is one such facility which lends a high performance language for technical computing.

The function oriented dataset considered have the output attribute as Reusability value. The Reusability in the dataset is expressed in terms of six numeric labels i.e. 1, 2, 3, 4, 5 and 6. The label 1 represents Nil and the label 6 represents the Excellent Reusability Label. The statistics of the count of the number of examples of certain reusability label is shown in the Table 4.1. The Graphical representation of the count of the number of examples of certain reusability label is shown in the Figure 4.1

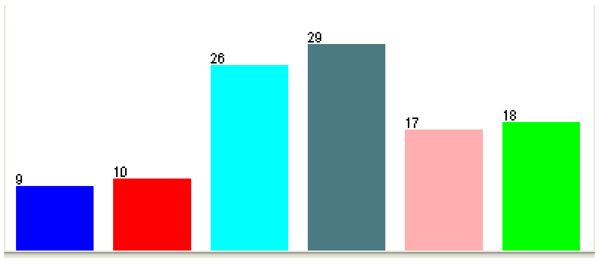
Table 4.1 statistics of the Reusability Output Attribute in the Dataset


Selected attribute

Name: Reusability　　Type: Nominal
Missing: 0 (0%)　　Distinct: 6　　Unique: 0 (0%)

No.	Label	Count
1	1	9
2	2	10
3	3	26
4	4	29
5	5	17
6	6	18

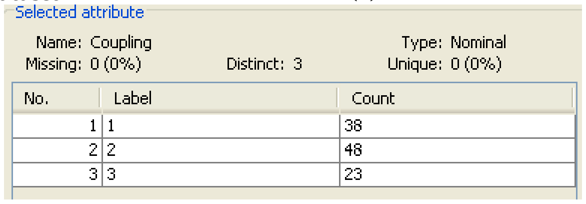
Figure 4.1 Bar-chart of Count of examples of the Reusability Output Attribute in the Dataset



The statistics shows that in the dataset, there are 9 examples of label 1, 10 examples of label 2, 26 examples of label 3, 29 examples of label 4, 17 examples of label 5 and 18 examples of label 6.

The input attribute-wise statistical details of the count of the examples of the labels are shown in table 4.2, table 4.3, table 4.4, table 4.5, table 4.6. The input attributes are expressed in the three linguistic labels i.e. 1, 2, and 3. The label 1 corresponds to the Low value, label 2 corresponds to the Medium value and label 3 corresponds to the high value.

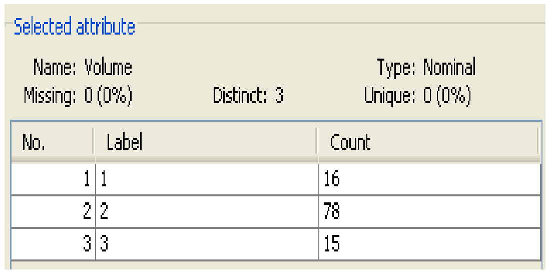
Table 4.2 statistics of the Input Attribute Coupling in the Dataset


Selected attribute

Name: Coupling　　Type: Nominal
Missing: 0 (0%)　　Distinct: 3　　Unique: 0 (0%)

No.	Label	Count
1	1	38
2	2	48
3	3	23

Table 4.3 statistics of the Input Attribute Volume in the Dataset

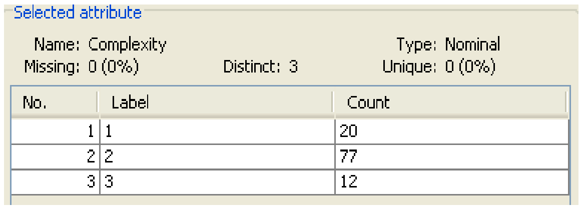


Selected attribute

Name: Volume | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

No.	Label	Count
1	1	16
2	2	78
3	3	15

Table 4.4 statistics of the Input Attribute Coupling in the Dataset

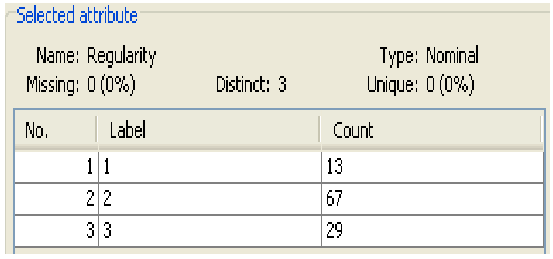


Selected attribute

Name: Complexity | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

No.	Label	Count
1	1	20
2	2	77
3	3	12

Table 4.5 statistics of the Input Attribute Coupling in the Dataset

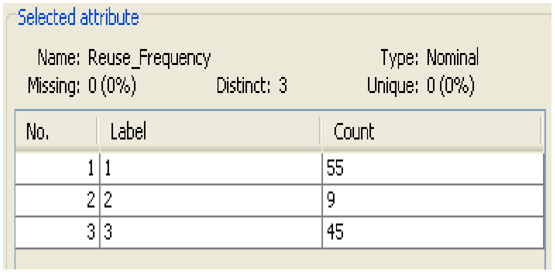


Selected attribute

Name: Regularity | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

No.	Label	Count
1	1	13
2	2	67
3	3	29

Table 4.6 statistics of the Input Attribute Reuse-Frequency in the Dataset



Selected attribute

Name: Reuse_Frequency | Type: Nominal
Missing: 0 (0%) | Distinct: 3 | Unique: 0 (0%)

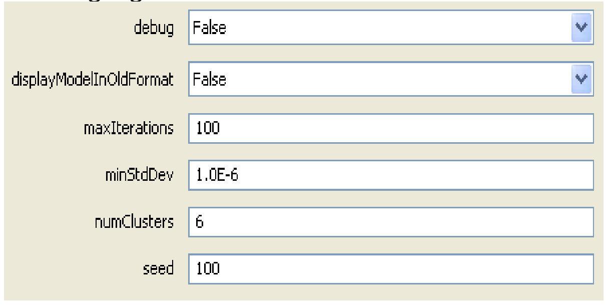
No.	Label	Count
1	1	55
2	2	9
3	3	45

The given data with five Input Attributes i.e. Coupling, Volume, Complexity, Regularity, Reuse_Frequency, and Output attributes is loaded in the Weka environment. First, the EM clustering ignores Reusability output attribute. EM assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters.
There following parameters are used in the algorithm:

- debug -- It is set to true so that formed clusters information can be displayed on the console.
- maxIterations – It is Maximum number of iterations. That are set to 100.
- minStdDev – It is Minimum allowable standard deviation. That is set to 1.0E-6 as default value.
- numClusters – It is the number of clusters. It is set to 6 as in the dataset there are six levels of reusability value.
- seed – It is random number seed to be used. It is set to 100 as default value.

The snapshot of the parameters set is shown in figure 4.2.

Figure 4.2. Snapshot of the Parameters Set in the EM Clustering Algorithm



debug: False
displayModelInOldFormat: False
maxIterations: 100
minStdDev: 1.0E-6
numClusters: 6
seed: 100

The EM clustering algorithm has created clusters numbered as 0 to 5 and assigned the 23 (means 21%) examples to cluster number 0, 24 (means 22%) examples to cluster number 1, 38 (means 35%) examples to cluster number 2, 18 (means 17%) examples to cluster number 3 and 6 (means 6%) examples to cluster number 4. The cluster 5 is not able to get any example. Further the cluster numbers are again assigned Predicted Labels as follows:

Table 4.7. The Assignment of Predicted Labels to the Clusters formed by EM

Cluster Number	**Predicted Label**
Cluster 0	6
Cluster 1	3
Cluster 2	4
Cluster 3	2
Cluster 4	1

The confusion matrix calculated is shown in Table 4.8.

Table 4.8 The Confusion Matrix Generated after applying EM Clustering

	Real Data Label					
Predicted Label	**1**	**2**	**3**	**4**	**5**	**6**

1	2	1	1	2	0	0
2	7	8	3	0	0	0
3	0	1	12	2	9	0
4	0	0	10	25	3	0
5	0	0	0	0	0	0
6	0	0	0	0	5	18

The Precision and Recall values for different the Reusability levels if the reusability is shown in table 4.9 and 4.10 respectively.

Table 4.9 Precision Value of Different Classes of the Reusability Values

Re usability Level Class	Precision Value
1	0.33
2	0.44
3	0.5
4	0.66
5	0
6	0.78

Table 4.10 Recall Value of Different Classes of the Reusability Values

Reusability Level Class	Recall Value
1	0.22
2	0.8
3	0.46
4	0.86
5	0
6	1

As evidenced from the confusion matrix the incorrectly clustered instances are 44.0 means 40.367 % is the inaccuracy value or correctly clustered instances are 65 means the accuracy is approximately 60%.

5 CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In this study Expectation Maximization based Clustering approach is evaluated for Reusability Prediction of Function based Software systems. Here, the metric based approach is used for prediction. Reusability value is expressed in the six linguistic values. Five Input metrics are used as Input and clusters are formed using EM, thereafter 10 fold cross validation performance of the system is recorded. As deduced from the results it is clear that Precision and Recall values of the sixth level reusability class is the maximum, it means the system is able to detect the "Excellent" components precisely. Similarly, Precision and Recall values of the fourth level reusability class is the second best, it means the system is able to detect the "Good" components with good precision.

The proposed technique is showing Accuracy value approximately equal to 60%, so it is satisfactory enough to use the Expectation maximization based clustering technique for the prediction of the function based reusable modules from the existing reservoir of software components.

5.2 Future Scope

The proposed approach is applied on the C based software modules/components and it can further be extended to the Artificial Intelligence (AI) based software components e.g. Prolog Language based software components. It can also be tried to calculate the fault-tolerance of the software components with help of the proposed metric framework.

The research work can be extended in the following directions:

- Intelligent Component Mining or Extraction algorithms can be developed
- Early prediction of the quality of component based system
- Characterization of Software Components for easy retrieval

6 ACKNOWLEDGMENTS

I would like to place on record my deep sense of gratitude to Dr. Himani Sharma, Dean P.G.Courses, BMSCE, Muktsar,, India for his generous guidance, and supervision throughout the course of present work. I express my sincere gratitude to Dr.D.S. Grewal, Director/Principal of Bhai Maha Singh College of Engineering, Muktsar India, for his stimulating guidance, continuous encouragement and useful suggestions

7 REFERENCES

1. Anderson, J.A (2003) "An Introduction To Neural Networks", Prentice Hall of India.
2. Arnold, R.S. (1990) "Heuristics for Salvaging Reusable Parts From Adav Code", SPC TechnicalReport, ADA_REUSE_HEURISTICS-90011-N, March 1990.
3. Arnold, R.S. (1990) "Salvaging Reusable Parts From Ada Code: A Progress Report", SPC Technical Report, SALVAGE_ADA_PARTS_PR-90048-N, September 1990.
4. Basili, V. R. and Rombach, H. D. (1988) "The TAME Project: Towards Improvement Oriented Software Environments", IEEE Trans. Software Eng., vol. 14, no. 6, June 1988, pp. 758-771.
5. Basili, V.R. (1989) "Software Development: A Paradigm for the Future", Proceedings COMPAC'89, Los Alamitos, California, IEEE CS Press, 1989, pp. 471-485.
6. Boetticher, G. and Eichmann, D. (1993) "A Neural Network Paradigm for Characterizing Reusable Software", Proceedings of the Australian Conference on Software Metrics, Australia, July, 1993, pp. 234-237.
7. Boetticher, G., Srinivas, K. and Eichmann, D. (1990) "A Neural Net-Based Approach to the Software Metrics" Proceedings of the 5th International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, 14-18 June 1990, pp. 271-274.

8. Caldiera, G. and Basili, V. R. (1991) “Identifying and Qualifying Reusable Software Components,” IEEE Computer, February 1991.

9. Chen, Y. F. Nishimoto, M. Y. and Ramamoorty, C. V. “The C Information Abstraction System”, IEEE Trans. on Software Engineering, Vol. 16, No. 3, March 1990.

10. Dunn, M. F. and Knight, J. C. (1993) “Software reuse in Industrial setting: A Case Study”, Proc. of the 13th International Conference on Software Engineering, Baltimore, MA, 1993. pp. 56-62.

11. Esteva, J. C. and Reynolds, R. G. (1991) “Identifying Reusable Components using Induction”, International Journal of Software Engineering and Knowledge Engineering, Vol. 1, No. 3 , 1991, pp. 271-292.

12. Frakes, W.B. and Kyo Kang (2005) “Software Reuse Research: Status and Future”, IEEE Trans. Software Engineering, vol. 31, issue 7, July 2005, pp. 529 - 536.

13. Jang, J-S. R. and Sun, C.T. (1995) “Neuro-fuzzy Modeling and Control”, Proceeding of IEEE, March 1995, pp. 123-135.

14. Jerome Feldman (1996) “Neural Networks - A Systematic Introduction” Berlin, New-York, 1996.

15. Kartalopoulos, S. V. (1996) “Understanding Neural Networks and Fuzzy Logic-Basic Concepts and Applications”, IEEE Press, 1996, pp. 153-160.

16. Klir, G. J. and Yuan, B. (1995) “Fuzzy Sets and Fuzzy Logic” Prentice-Hall, New Jersey.

17. Mayobre, G. (1991) “Using Code Reusability Analysis to Identify Reusable Components from Software Related to an Application Domain,” Proceeding of the Fourth Workshop on Software Reuse, Reston. VA, November, 1991, pp. 87-96.

18. Nguyen, H.T. and Walker, E.A. (1997) “A first course in Fuzzy Logic” Boca Raton FLA. CRC Press, 1997.

19. Melanie Mitchell (1996) “An Introduction to Genetic Algorithm”, MIT Press, 1996.

20. Parvinder Singh and Hardeep Singh (2005) “Critical Suggestive Evaluation of CK METRIC”, Proc. of 9th Pacific Asia Conference on Information Technology (PACIS-2005), Bangkok, Thailand, July 7 – 10, 2005, pp 234-241.

21. Poulin, J. S. (1997) “Measuring Software Reuse– Principles”, Practices and Economic Models, Addison-Wesley, 1997.

22. Richard, W. S. (2005) “Enabling Reuse-Based Software Development of Large-Scale Systems”, IEEE Trans. on Software Engineering, Vol. 31, No. 6, June 2005 pp. 495-510.

23. Selby, R. W. (1988) “Empirically Analyzing Software Reuse in a Production Environment”, Software Reuse: Emerging Technology, W. Tracz, ed, IEEE Computer Society Press, 1988.

24. Stender (1994) “Introduction to genetic algorithms”, IEEE Colloquium on Genetic Algorithms, Volume 2, March 15, 1994 pp. 1-4.