Journal of Computer Science 6 (10): 1078-1082, 2010
ISSN 1549-3636
© 2010 Science Publications

Pattern Trees for Fault-Proneness Detection in Object-Oriented Software

[1]Romana Ishrat, [2]Rafat Parveen and [3]Syed I. Ahson
[1]Centre for Interdisciplinary Research in Basic Sciences,
[2]Department of Computer Science,
Jamia Millia Islamia, New Delhi-25, India
[3]Department of Administration, Patna University, Patna-800-005, India

Abstract: Problem statement: This study introduced an application of pattern tree based classification technique in the area of object-oriented software quality estimation. This application explored the fault prediction accuracy of pattern trees. **Approach:** Similarity measures and fuzzy aggregations employed in the pattern tree technique had been used to generate tree models for fault detection in software modules. Experiments had been performed on datasets namely, KC1 and KC3 obtained from NASA's metric data program. Pattern tree models were built using metrics from the object-oriented software datasets. **Results:** AND/OR, OWA and WA had been selected for pattern tree induction. Pattern tree models build using RMSE similarity measure produced higher accuracy as compared to other similarity measures. **Conclusion:** The proposed application succeeded in improving the quality of the object-oriented software in terms of prediction accuracy. Pattern trees models were found to be less structural complex as compared to fuzzy decision trees.

Key words: Pattern trees, object-oriented software, fault prediction accuracy, quality estimation

INTRODUCTION

Advances in distributed object technologies dramatically impact the development process of distributed software applications. In particular, time for providing new distributed services is decreasing because applications are not built from scratch any longer (Denaro *et al*., 2003). Object-oriented technology brings great ease in software redevelopment areas. One of the new issue is that how to develop quality of the system and how to measure and improve software quality for both development and re-development. Object-oriented design plays a pivotal role in software development, because it determines the structure of the software solution (Khan *et al*., 2006). Software quality estimation is a key factor in developing a software system. High-assurance software systems depend on the stability and reliability of underlying software. The goal of software quality estimation is intangible in an actual project environment. The quality cannot be directly checked in the product, it must be planned right from the beginning. The failure rate of software is high in the early stage of software development life cycle, due to the undiscovered errors or faults. Software faults are common reasons of complexity in modern systems. Software faults are the defects that cause a software failure in an executable product (Khoshgoftaar and Seliya, 2002).

A lack of quality in design process can make correct implementation impossible. If these faults are not found earlier in object-oriented software modules then it will be very costlier to fix them in the end thereby decreasing the quality of the end product. Finding faults in the early stage increases the quality of the end product and prevent ripple effects from the changes later in the software development life cycle. It is wise to isolate the faults as early as possible in design phase. Therefore estimation of quality of software has become an important factor in software development. Software metrics have become essential in software engineering for several reasons, among which quality assessment and reengineering. In the field of software evolution, metrics can be used for identifying stable or unstable parts of software systems (Lanza and Ducasse, 2002). Software metrics is a necessary step for quality and reliability (Wang *et al*., 1997).

Decision tree is one of the simplest software quality modeling techniques used in software quality estimation (Ishrat *et al*., 2009). Decision tree is one of

Corresponding Author: Romana Ishrat, Centre for Interdisciplinary Research in Basic Sciences, Jamia Millia Islamia, New Delhi-25, India Tel: +919891813458

the most widely used practical methods for inductive inference (Mitchell, 1997). Software quality estimation models have been built using various decision tree techniques. Khoshgoftaar and Seliya (2002) and Wang *et al*. (1997) have applied regression tree algorithms for software fault prediction. Khoshgoftaar and Seliya (2002) and Khan *et al*. (2006) have also applied classical decision tree algorithms like C4.5, CART and S-Plus for software quality estimation. These models effectively minimized software failures and improved the reliability of the software systems. Classical decision trees and ensemble techniques (Ishrat *et al*., 2009), fuzzy decision trees technique (Ishrat *et al*., 2010) have been applied to build quality estimation models for object-oriented software data.

Pattern trees: Like decision trees, pattern trees are an effective tool for classification applications. A novel pattern tree induction method has been proposed to build the pattern trees, by means of the similarity measures and different aggregation operators (Huang and Gedeon, 2006). A pattern tree is used to represent pattern of data which belong to the same class. Under binary context, the fact that a data matches a given pattern tree induces that the data should be classified into the class that the pattern tree represents. Under fuzzy context, the matches of a data and a given pattern tree would not be crisp yes or no, instead, a truth value which is in the range of [0, 1] is obtained to reflect how confident a data should be classified to the class that the pattern tree represents (Huang, 2007). A pattern tree is a tree which propagates fuzzy terms using different fuzzy aggregations. Each pattern tree represents a structure for one output class which is located at the top as the root of the tree. The pattern tree induction methods are based on similarity measures and fuzzy aggregations. Note that all the nodes within the pattern trees are leaf nodes. When a new data sample is tested over a pattern tree, it starts from the bottom leaves and travels to the top. It finishes with a truth value indicating the degree of this data sample belonging to the output class of this pattern tree. The output class with the maximal truth value is chosen as the prediction class (Huang and Gedeon, 2006).

Similarity measures: Let A and B be two fuzzy sets (Zadeh, 1965) defined on the universe of discourse U. The commonly used fuzzy similarity definitions are shown in Table 1, where $\cap$ and $\cup$ denote a certain t-norm operator and a t-conorm respectively. The fuzzy similarity (Chao *et al*., 1996) between them can be defined as:

$$S(A,B) = \frac{A \cap B}{A \cup B} \tag{1}$$

where, $\cap$ and $\cup$ denote a certain t-norm operator and a t-conorm respectively. Usually, the MIN($\wedge$) and MAX($\vee$) operators are used. According to the definition, $0 \le S(A,B) \le 1.$ in practice, this measurement can be computer as:

$$S(A,B) = \frac{\sum_{j=1}^{m} [\mu_A(x_j) \wedge \mu_B(x_j)]}{\sum_{j=1}^{m} [\mu_A(x_j) \vee \mu_B(x_j)]} \tag{2}$$

Where:

$x_j, j = 1,....,m,$	=	The crisp values are discredited in the variable domain
$\mu_a(x_j) \text{and} \mu_B(x\ \ )$	=	The fuzzy membership values of x for A and B

An alternate similarity definition proposed by Huang and Gedeon (2006) and Huang *et al*. (2008) for pattern tree construction is Root Mean Square Error (RMSE) based fuzzy set similarity. Consider that the Root Mean Square Error of fuzzy sets A and B can be compared as:

$$\text{RMSE}(A,B) = \sqrt{\frac{\sum_{j=1}^{m} (\mu_A(x_j) - \mu_B(x_j))^2}{m}} \tag{3}$$

The RMSE based fuzzy set similarity can be defined as:

$$\text{Sim}(A,B) = 1 - \text{RMSE}(A,B) \tag{4}$$

The large value Sim(A, B) takes, the more similar A and B are.

Fuzzy aggregations: Fuzzy aggregations are logic operators applied to fuzzy membership values or fuzzy sets. They have three sub-categories, namely t-norm, t-conorm and averaging operators such as Weighted Averaging (WA) and Ordered Weighted Averaging (OWA) (Huang and Gedeon, 2006; Yager, 1988). In fuzzy sets theory, triangular norms (t-norm) and triangular-conorms (t-conorm) are extensively used to model logical operators *and* and *or*. The basic t-norm and t-conorm pairs which operate on two fuzzy membership values a and b, $a, b \in [0,1]$ are shown in Table 1.

Table 1: Basic t-norms and t-conorms pairs

Name	T-norm	T-conorm
MIN/MAX	min{a, b} = $a\wedge b$	miax{a, b} = $a\vee b$
Algebraic AND/OR	ab	a+b-ab
Lukasiewicz	max{z+b-1,0}	min{a+b–1}
EINSTEIN	$\frac{ab}{2-(a+b-ab)}$	$\frac{a+b}{1+ab}$

The aggregations above are only shown to apply to a pair of fuzzy values; they can also be applied to multiple fuzzy values as they retain associatively.

A WA operator of dimension n is a mapping $E:R^n \to R$, that has an associated n-elements vector $w=(w_1,w_2,....w_n)^T, w_i \in [0,1], 1\le i\le n$ and $\sum_{i=1}^{n} w_i = 1$ so that $E(a_1.....a_n)=\sum_{j=1}^{n} w_j a_j$.

An OWA operator (Yager, 1988) of dimension n is a mapping $F:R^n \to R$, that has an associated n-elements vector, $w=(w_1,w_2,....w_n)^T, w_i \in [0,1], 1\le i\le n$ and $\sum_{i=1}^{n} w_i = 1$ so that $F(a_1.....a_n)=\sum_{j=1}^{n} w_j b_j$ where b_j is the jth largest element of the collection $\{a_1,\dots,a_n\}$.

The fundamental difference of the OWA from WA aggregation is that the former does not have a particular weight w_i associated for an element, rather a weight is associated with a particular ordered position of the element. The main factor to determine which aggregation should be used relies on the relationship between the criteria involved (Huang and Gedeon, 2006).

MATERIALS AND METHODS

Datasets: The empirical software datasets used in the case study have been taken from NASA IV and V Facility Metrics Data Program, a freely available repository website. This repository contains software metrics and associated error data. The two datasets namely, KC1 and KC3 have been used contains a set of software metrics and an additional attribute called fault, to check whether a module is faulty or not. The fault prone modules constitute only small portion in the datasets (NASA, 2008). The numbers of cases collected in these datasets belong to one of the two classes either faulty or non-faulty. Each dataset contains different number of software metrics. The metrics involved in the datasets were taken as independent variable. The dependent variable is fault or non-fault modules. Table 2 shows the description of the datasets.

Table 2: Datasets used in the experiments

Project	Language	Modules	Metrics	Defects (%)	Description
KC1	C++	2107	26	15.5	Storage management for processing and receiving ground data
KC3	Java	458	42	6.3	Processing and delivery of satellite metadata

```
1.  Comment: Initialization
2.  $P=\{A_{ij}\}, i=1,\dots,n; j=1,\dots,m$
3.  $C_0=\arg\max_{P\in P}[Sim(P,X_0)]$
4.  Comment: Induction
5.  for k=1 to d step 1 do
6.  $\{\Psi_k,S_k\}=\arg\max_{\substack{\Psi\in\Psi,S\in P\\ S\notin C_{k-1}}}[Sim(C_{k-1}\Psi S,X_0)]$
7.  $C_k=C_{k-1\Psi_n}S_k$
8.  if $Sim(C_k,X_0)<Sim(C_{k-1},X_0)$ then
9.  k=k-1
10. break
11. end if
12. end for
13. return $C_k$
```

Fig. 1: Induction of simple pattern tree

Data preprocessing: These datasets have been preprocessed to a format acceptable by the pattern tree software tool, before they are used in the experiments. For all datasets, a simple fuzzification method based on three evenly distributed trapezoidal membership functions for each input variable i.e., metrics from the datasets is used to transform the crisp to fuzzy values (Huang, 2007). The whole datasets are divided into training and test sets.

Pattern tree induction method: Assume a dataset has n input variables A_i, = 1,2,…,n and one output variable X. Further assume that input variables each have m fuzzy linguistic terms denoted as A_{ij}, i = 1,2,…,m and output variable has k fuzzy or linguistic terms denoted as x_j, = 1,2,…,k. That is, each data point is represented by a fuzzy membership value vector of dimension (nm+k). The task is to build k pattern trees for the k output classes (fuzzy or linguistic terms). The task is to build k pattern trees for the k output classes (fuzzy or linguistic terms). The induction of pattern tree, say for class X_0, is described in algorithm shown in Fig. 1. The induction of other pattern trees follows the same principle.

In the initialization, the set of primitive trees P is constructed, in which each fuzzy term A_{ij}, i = 1,…,n, j = 1,…,m is use to construct a primitive pattern tree.

The primitive tree which has highest similarity to output class term X_0, is then selected as the initial candidate tree C_0. Here P indicates that it contains a set of trees in contrast to one tree such as C_0. The subscript of zero in C_0 indicates that tree has zero depth. In induction, the aggregation is attempted between the previous candidate tree C_{k-1} and any primitive tree S in the primitive tree set P, using any aggregation ψ drawn from the aggregation set ψ. When ψ = WA or ψ = OWA, the weights which make the aggregated term most similar to class term used. A constraint is imposed upon the aggregation: The primitive tree S cannot be a subset tree of the candidate tree C_{k-1}, which prevents a primitive tree being used in the aggregated tree more than once. Among all aggregated trees, the one which has the highest similarity to class term X_0 is selected as the current candidate tree C_k, which has one more depth than the previous candidate tree C_{k-1}. If the candidate tree has reached the pre-defined depth d, or the new candidate tree C_k has a lower similarity to X_0 than the previous one C_{k-1}, the induction stops and the tree which has the highest similarity is returned as the optimal tree. In this algorithm, an aggregation always happens between a candidate tree and a slave primitive tree. The aggregated trees thus always have one fuzzy term as its right child for the internal node. This kind of tree is denoted as simple pattern trees. In contrast, pattern trees which do not have such a constraint is referred to as general pattern trees (Huang, 2007).

RESULTS AND DISCISSION

The experiments have been carried out using KC1 and KC3 datasets. The aim is to estimate the quality of the object-oriented software by predicting the number of faults. Pattern tree models were built using all the software metrics from the two data sets. Out of all aggregations mentioned above and /OR, OWA and WA have been selected for pattern tree induction. RMSE and Jaccard similarity measures are tried on the both datasets, out of which RMSE produced promising results. The maximum depth d is set to 3 and the candidate tree level is 2. The performance of both datasets is shown in Table 3.

In Fig. 2 FTerm0 and FTerm1 are the fuzzy terms associated with their respective input variables i.e., the metrics. The oval shapes are input variables and the number inside these oval shapes denote the following metrics participated in pattern tree induction:

- ERROR_REPORT_IN_1_YR
- HALSTEAD_LEVEL
- ERROR_DENSITY
- NUM_OPERANDS

Table 3: Prediction accuracy of pattern tree

Pattern tree	KC1	KC3
Prediction accuracy	96.51%	95.80%

Table 4: Prediction accuracy of pattern tree and fuzzy decision tree

Data sets	Pattern tree (%)	Fuzzy decision tree (%)
KC1	96.51	96.40
KC3	95.80	95.50

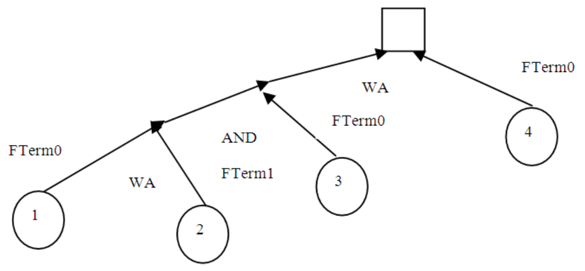

Fig. 2: Pattern tree for class 0 of KC1 dataset

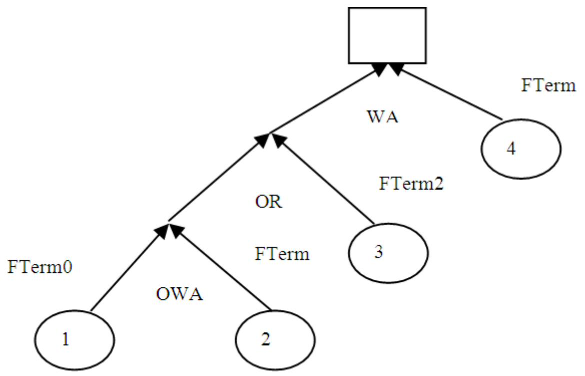

Fig. 3: Pattern tree for class 1 of KC1 dataset

In Fig. 3 the following metrics corresponds to the numbers inside the oval shapes:

- ERROR_COUNT
- ERROR_COUNT
- ERROR_COUNT
- LOC_CODE_AND_COMMENT

The performance of the proposed application is evaluated and compared with the fuzzy decision tree (Ishrat *et al*., 2010) models. The prediction accuracy of the pattern trees and the fuzzy decision trees are shown in Table 4. It can be observed that pattern trees performed in a consistent way for both datasets. The pattern tree results in higher classification accuracy than fuzzy decision tree. Structural complexity of pattern trees is less than fuzzy decision trees.

CONCLUSION

This study has proposed a new application of decision tree termed pattern trees, which make use of different aggregations including both t-norms and t-conorm, for quality estimation in the area of object oriented software. Like decision trees, pattern trees are found to be an effective tool for classification applications. The pattern tree induction methods are based on similarity measures such as RMSE and fuzzy aggregations OWA and WA. The pattern trees have been generated for faults prediction in the software modules using all the metrics from the datasets. The pattern trees build using RMSE similarity measure produced best results. The pattern trees performed consistently. The comparison to fuzzy decision tree shows that the pattern tree can obtain higher classification accuracy. The pattern trees are found to be less complex in structure than fuzzy decision trees.

REFERENCES

Chao, C.T., Y.J. Chen and C.C. Teng, 1996. Simplification of fuzzy neural systems using similarity analysis. IEEE Trans. Syst. Man Cybernet. Part B Cybernet., 26: 344-354. DOI: 10.1109/3477.485887

Denaro, G., L. Lavazza and M. Pezze, 2003. An empirical evaluation of object oriented metrics in industrial setting. J. Object Technol., 4: 1-4.

Huang, Z. and T.D. Gedeon, 2006. Pattern trees. Proceeding of the IEEE International Conference on Fuzzy Systems. Sept. 2006, WCCI, Vancouver, BC, Canada, pp: 1784-1791.

Huang, Z., 2007. Pattern tree software (fuzzy-EBY) user guide. EECS. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.126.6444

Huang, Z., T.D. Gedeon and M. Nikravesh, 2008. Pattern trees induction: A new machine learning method. IEEE Trans. Fuzzy Syst., 16: 958-970. DOI: 10.1109/TFUZZ.2008.924348

Ishrat, R., R. Parveen and S.I. Ahson, 2009. Decision tree techniques for object-oriented software quality estimation. Proceeding of the International Conference on Data Management, Feb. 2009, Institute of Management Technology, Ghaziabad, pp: 288-294.

Ishrat, R., R. Parveen and S.I. Ahson, 2010. Object-oriented software quality estimation-a fuzzy decision tree perspective. Int. J. Math. Sci. Eng. Appli., 4: 33-46.

Khan, R.A., K. Mustafa and S.I. Ahson, 2006. Software Quality: Concepts and Practices. 1st Edn., Narosa Publications House, New Delhi, ISBN: 9788173197222, pp: 216.

Khoshgoftaar, T.M., N. Seliya, 2002. Tree based software quality estimation models for fault prediction. Proceedings of the 8th IEEE Symposium on Software Metrics, June 4-7, IEEE Computer Society, Washington DC., USA., pp: 203-203. DOI: 10.1109/Metric.2002.1011339

Lanza, M. and S. Ducasse, 2002. Beyond language independent object oriented metrics: Model independent metrics. Proceedings of 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, (QAOOSE'02), BIBTEX, Malaga, pp: 77-84.

Mitchell, T.M., 1997. Machine Learning. 1st Edn., McGraw Hill, New York, ISBN: 10: 0070428077, pp: 432.

NASA, 2008. Metrics data repository. NASA. http://www.mdp.ivv.nasa.gov

Wang, Y.H., C.M. Chung, T.K. Shih, H.C. Keh and W.C. Lin, 1997. Object-oriented software quality through data scope complexity measurement. Proc. IEEE International Conference on Computational Cybernetics and Simulation, Oct. 12-15, IEEE Xplore Press, Orlando, FL., pp: 3849-3854. DOI: 10.1109/ICSMC.1997.633271

Yager, R.R., 1988. On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Trans. Syst. Man Cybernet., 18: 183-190. DOI: 10.1109/21.87068

Zadeh, L.A., 1965. Fuzzy sets. Inform. Control, 8: 338-353. DOI: 10.1016/S0019-9958(65)90241-X