An Over-sampling Method for Analogy-based Software Effort Estimation

Yasutaka Kamei[1], Jacky Keung[2], Akito Monden[1], Ken-ichi Matsumoto[1]

[1] Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{yasuta-k, akito-m, matumoto}@is.naist.jp

[2] National ICT Australia Ltd., Australian Technology Park, Sydney 1430 NSW, Australia
Jacky.Keung@nicta.com.au

ABSTRACT

This paper proposes a novel method to generate synthetic project cases and add them to a fit dataset for the purpose of improving the performance of analogy-based software effort estimation. The proposed method extends conventional over-sampling method, which is a preprocessing procedure for n-group classification problems, which makes it suitable for any imbalanced dataset to be used in analogy-based system. We experimentally evaluated the effect of the over-sampling method to improve the performance of the analogy-based software effort estimation by using the Desharnais dataset. Results show significant improvement to the estimation accuracy by using our approach.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *Cost estimation*

General Terms

Management, Measurement, Experimentation

Keywords

Software effort estimation, analogy, over-sampling, empirical study

1. INTRODUCTION

Software effort estimation by Analogy has been proposed and studied to support project resources planning and control [5], [6], [7]. This method derives an effort estimate from a number of project features, such as functional size measures, manager's skill level and software development environment. The procedure of analogy-based method can be summarized in three steps. First, the project features of a target project are given as the input. Next, one or more past analogous projects that are similar to the target project are identified from a completed projects data repository. Finally, the effort estimate for the target project is then derived based on the actual effort values of the analogous projects.

In most of the cases, the number of available past completed projects that have similar project characteristics with respect to the target project is small, subsequently the estimation accuracy is likely to be reduced. Figure 1(a) illustrates an example of an imbalanced dataset where the number of projects with large functional size is relatively small. The distances to the nearest neighboring projects are large and therefore these nearest neighboring projects becomes less useful for the estimation purpose. The objective in here is to provide a mechanism to minimize the distances to the nearest neighboring projects.

In this paper, we propose a novel approach to generate synthetic project cases and add them to a fit dataset. The approach described in this paper is called over-sampling method that adds synthesized projects similar to triangular plots shown in Figure 1(b). As shown in Figure 1(b), the synthetic projects like triangular plots are added in the middle of functional size to large of functional size, and then the number of projects that are similar features with target project would be increased. Hence, the over-sampling could have potential for the performance improvement of the analogy-based method. However, to our knowledge, no study has reported the effects of applying the over-sampling to general effort estimation. Mittas et al. has proposed "resampling" such as iterated bagging method [6] that selects some projects from all projects and uses them as a new dataset. On the other hand, "over-sampling" means the addition of a new synthetic project based on characteristics of projects in a dataset. We have experimentally evaluated the effects of applying the over-sampling method using the Desharnais dataset from a Canadian Software house.

Section 2 introduces the over-sampling method. Section 3 describes the details of the evaluation setting. Section 4 provides the result and discussion of the experiment. Section 5 summarizes the paper and presents some future directions of research.

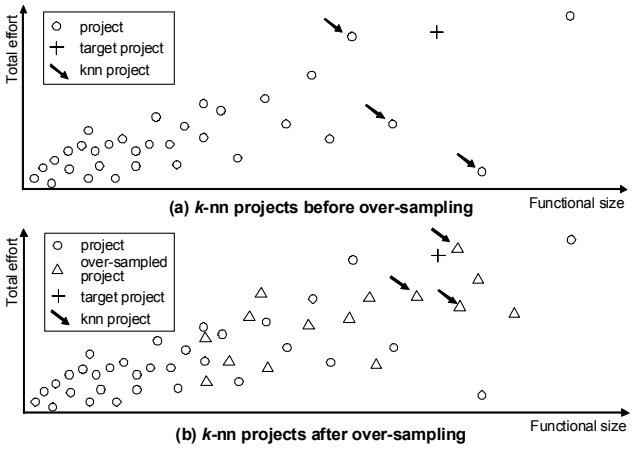

Figure 1. Examples of *k*-nearest neighboring projects before and after over-sampling

ESEM'08, October 9–10, 2008, Kaiserslautern, Germany.

2. OVER-SAMPLING METHOD

An over-sampling method generates synthetic cases based on the existing cases in a fit dataset. The conventional over-sampling method, which is suitable to n-group classification problems[1], can equalize the number of cases of each group in a fit dataset so as to improve the accuracy of classification models built from the dataset. In a similar study, Kamei et al. have shown that the application of the over-sampling methods on fault-prone module prediction has been promising [4].

This paper extends the over-sampling method for the cases in the project dataset where its dependent variable (such as effort) is continuous. First, we calculate the Pearson correlation between the dependent variable (effort) and each of the independent variables. Then, one the independent variables having the largest correlation is divided into x equal interval parts (in this paper, x is set to 3). For instance, if the functional size has the largest correlation to the effort, its minimum is 60 and the maximum is 780 in the dataset, it is divided into 3 parts as follows: [60, 300), [300, 540), [540, 780]. Then, we repeat the steps of the conventional over-sampling method for each part until the largest number of projects among all parts becomes equal to the number of projects in each part. In this paper we use the Synthetic Minority Over-sampling Technique SMOTE[1], a well-known over-sampling method. SMOTE generates synthetic project cases based on the k-nearest neighbors and the steps of SMOTE are the following.

Step 1. Selection of a minority case
One case (in this paper, a project) denoted as m_a is selected from a minority class in a dataset.

Step 2. Identification of k-nearest neighbors
K-nearest neighbors of m_a are identified based on the similarity computation. Below describes how to compute the similarity.

Step2-1. The normalization of predictor variables
The normalized value $norm(v_{i,j})$ of variable f_j of case m_i is calculated by the following equation:

$$norm(v_{i,j}) = \frac{v_{i,j} - \min(f_j)}{\max(f_j) - \min(f_j)} \quad (1)$$

where $\min(f_j)$ and $\max(f_j)$ denote maximum and minimum value in variable f_j , respectively.

Step 2-2. Similarity calculation
The similarity between ma and all other cases in the minority class is calculated. In this paper, we used Euclidian distance and cosine similarity, which are widely used as similarity measure.

Step 3. Selection of a neighbor
One case m_r was selected randomly from the k-nearest neighbors.

Step 4. Addition of a minority case
A new case was added in a place along a straight line between the vertexes of the feature vector of cases m_a and m_r.

3. EXPERIMENT

3.1 Overview

In this experiment, we are over-sampling the dataset before applying analogy to estimate effort using the Desharnais dataset. The estimation result is then compare with the result derived without over-sampling. Each independent variable is normalized based on equation (1) before applying analogy. We use the Euclidean distance to compute the similarity between the target project and each of the completed projects, and an effort estimate can be derived by averaging the effort of k-nearest neighboring (k-nn) projects. Note that k is set to 3.

We use conventional performance measures, such as the mean magnitude of relative error (MMRE), and Pred25[5], [7]. MMRE is the degree of estimation error against the actual effort. Smaller MMRE value indicates better estimation accuracy. Pred25 is defined as the percentage of predictions falling within 25% of the actual known value. Conversely, a larger Pred25 value indicates better estimation performance.

3.2 Dataset

The Desharnais dataset is used in this study. This dataset comprises 77 completed software project data from a Canadian Software house. It was first reported in Desharnais[2] and was used in the early study of analogy by Shepperd and Schofield [7]. The project size ranges from 62 FPs to 1116 FPs and the project effort ranges from 546 person hours to 23940 person hours.

The Desharnais dataset was then divided on the basis of differing development environments, because it is unlikely that a company would have access to such large volumes of data, and because smaller, more homogenous datasets are more useful for effort estimation. This dataset grouping approach is similar to the study in [5], [7] and, where the Desharnais dataset is divided into Desharnais-1(44cases), Desharnais-2(23 cases) and Desharnais-3(10 cases).

3.3 Experimental Procedure

In this paper, each estimated effort was evaluated by leave-one-out cross validation. The experimental procedure is as follows.

1. Select one project from a dataset as the target project *test*. The others are the completed projects *fit*.
2. Apply our over-sampling method to *fit*. The resultant dataset is *fit'*.
3. Compare the estimated effort of *test* based on *fit'* to the actual effort of *test*.
4. Repeat steps 1 to 3 for all projects in the dataset.

4. RESULT AND DISCUSSION

The performances of the analogy-based effort estimation with the over-sampling method and without that are shown in Table 1. The column "% improved" shows the improvement compared to the no sampling, and the value "-" in the column shows the deteriorated performance. As shown in Table 1, the performance of the over-sampling with cosine similarity was better than that of the over-sampling with Euclidean distance in the all datasets. In the case of the over-sampling with cosine similarity, the improvement of MMRE was 3.1% and that value of Pred25 was 17.6% in Desharnais-1. Also, in Desharnais-2, MMRE was slightly decreased (1.8%), but Pred25 was considerably improved (37.5%). On the other hand, both of MMRE and Pred25 were decreased 1.2% and 20% in Desharnais-3.

In addition to Table 1, the estimation performance (MRE and MAE) of the top 5 large effort projects is shown in Figure 2. The numbers in x-axis indicate the rank of size of actual effort in each dataset. The gray bar indicates the case without over-sampling

Table 1. Estimation performance of each method

Dataset	No sampling	Sampling with normalized Euclid distance		Sampling with normalized cosine similarity	
	MMRE	MMRE	%improved	MMRE	%improved
Desh 1.	0.473	0.468	1.049	0.458	3.121
Desh 2.	0.393	0.440	- 11.929	0.400	- 1.845
Desh 3.	0.488	0.515	- 5.686	0.494	- 1.283
	Pred(25)	Pred(25)	%improved	Pred(25)	%improved
Desh 1.	0.386	0.409	5.882	0.455	17.647
Desh 2.	0.348	0.435	25.000	0.478	37.500
Desh 3.	0.500	0.400	- 20.000	0.400	- 20.000

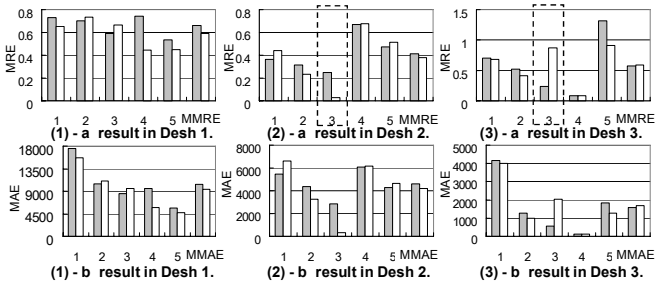

Figure 2. Estimation performance for the top 5 projects of size of actual effort

and the white bar indicates the case with over-sampling. The improvement of MMRE was 10.7% in Figure 2(1)-a and that of MMRE was 8.5% in Figure 2(2)-a. This means that the estimation performance of large effort projects was improved in Desharnais-1 and 2. On the other hand, the estimation performance of the project 3 in Figure 2(3)-a was heavily decreased by 270.0%. For this reason, MMRE for Desharnais-3 was decreased by 3.2%.

Further analysis in Figure 3 shows *k*-nn projects including the project 3 in Desharnais-2 and the project 3 in Desharnais-3, which are framed in by dotted rectangles in Figure 2. The project 3 in Desharnais-2 is the best improved project by over-sampling, and the project 3 in Desharnais-3 is the worst project by over-sampling. In Figure 3, x-axis indicates the adjusted function point (Adj.FP), which is an independent variable that had the largest correlation to the actual effort. In Figure 3(a), by applying over-sampling, the distances from target project (proj. 3) to the *k*-nn projects became small. We believe that this contributed to the improvement of performance. On the other hand, in Figure 3(b), the distances to the *k*-nn projects became small as well, however, there was no improvement. As shown in Figure 3(b), the actual effort of the target project was extremely small although its Adj.FP was large, that is, this project could be considered an outlier, Therefore, the estimation performance of Desharnais-3 in Table 1 and Figure 2 could be decreased.

5. CONCLUSION

In this paper, we argue that applying the over-sampling method to analogy-based software cost estimation will improve its prediction accuracy, and the effect has been experimentally evaluated. The result shows that the performance improvements as indicated by MMRE and Pred25 are 3.1% and 37.5% respectively at their maximum. And, for projects with large effort, the performance improvement as indicated by MMRE was 10.7% at its maximum.

The result of this work is encouraging and we are anticipating more experimental trials of our method on other datasets. We will also consider using other performance measure other than MMRE in future experimentation, as MMRE is often considered unreliable[3].

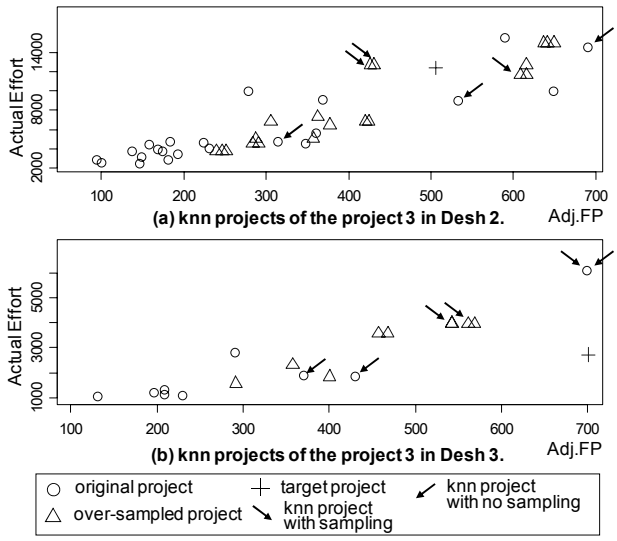

Figure 3. Examples of *k*-nn projects in the experiment

6. ACKNOWLEDGMENTS

This work is being conducted as a part of StagE Project, the Development of Next Generation IT Infrastructure, supported by Ministry of Education, Culture, Sports, Science and Technology and Grant-in-Aid for Japan Society for the Promotion of Science (JSPS) Fellows (Research No:20009220). Special thanks go to empirical software engineering research program's members in NICTA, especially Prof. Ross Jeffery, for their helpful comments on this study.

7. REFERENCES

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16:321-357, 2002.

[2] J. M. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. Master's thesis, University of Montreal, 1989.

[3] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion MMRE. IEEE Trans. Software Engineering, 29(11):985-995, 2003.

[4] Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto, and K. Matsumoto. The effects of over and under sampling on fault-prone module detection. In Proc. Int'l Symposium on Empirical Software Engineering and Measurement (ESEM'07), pages 196-204, 2007.

[5] J. W. Keung and B. Kitchenham. Optimising project feature weights for analogy-based software cost estimation using the mantel correlation. In Proc. Asia-Pacific Software Engineering Conference(APSEC'07), pages 222-229, 2007.

[6] N. Mittas, M. Athanasiades, and L. Angelis. Improving analogy-based software cost estimation by a resampling method. Information and Software Technology, 50(3):221-230, 2008.

[7] M. Shepperd and C. Schofield. Estimating software project effort using analogies. IEEE Trans. Software Engineering, 23(11):736-743,1997.