Segmentation of Software Engineering Datasets Using the M5 Algorithm

D. Rodríguez[1], J.J. Cuadrado[2], M.A. Sicilia[2], and R. Ruiz[3]

[1] The University of Reading,
Reading, RG6 6AY, UK
d.rodriguezgarcia@rdg.ac.uk
[2] The University of Alcalá,
28805 - Alcalá de Henares (Madrid), Spain
{jjcc, msicilia}@uah.es
[3] The University of Seville,
41012 - Avda Reina Mercedes s/n., Sevilla, Spain
rruiz@rdg.ac.uk

Abstract. This paper reports an empirical study that uses clustering techniques to derive segmented models from software engineering repositories, focusing on the improvement of the accuracy of estimates. In particular, we used two datasets obtained from the International Software Benchmarking Standards Group (ISBSG) repository and created clusters using the M5 algorithm. Each cluster is associated with a linear model. We then compare the accuracy of the estimates so generated with the classical multivariate linear regression and least median squares. Results show that there is an improvement in the accuracy of the results when using clustering. Furthermore, these techniques can help us to understand the datasets better; such techniques provide some advantages to project managers while keeping the estimation process within reasonable complexity.

Keywords: Effort estimation, Data mining, Tress, M5.

1 Introduction

Effort estimation is one of first and more important activities that project managers face at the beginning of each project. Accurate estimates are an integral part of all process improvement activities. The current state-of-practice for effort and cost estimates entails the use one of the many public (e.g., COCOMO [2]) or private (e.g., PRICE S [13]) parametric models. Both public or commercial models use classical regression as their foundation for deriving equations from historical project databases. During the last decade, several organizations such as the International Software Benchmarking Standards Group (ISBSG) [10] have started to collect project management data from a variety of organizations. In this way, companies without historical datasets could use these generic databases for estimation or companies already collecting data could compare themselves

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part IV, LNCS 3994, pp. 789–796, 2006.

with other industries, i.e., benchmarking. One problem faced by project managers using such databases is the heterogeneity of the projects (e.g., the latest release of the ISBSG has more than 50 attributes and 3000 instances). Specifically, heterocestacity (non-uniform variance) is known to be a problem affecting datasets that combine data from heterogeneous sources. For example, a straightforward application of least square regression using the 709 projects, which can be obtained from the Reality estimation tool accompanying the ISBSG repository, results in measures of median relative error of 280% and less than 23% of the estimates are within the 75% of actual values. This leads to poor estimates in the software engineering. In this paper, we describe an empirical study on the appropriateness of model trees, i.e., a decision tree with a linear regression model for each subset, with relation to predictive quality. Concretely, we derived several datasets from the ISBSG repository and also compare the outputs of using M5 with multivariate Linear Regression (LR) and Least Median Squares (LMS) .

This paper is organized as follows. Section 2 and 3 describe respectively the methods of analysis and evaluation techniques used in this paper. Section 4 describes the datasets used for this analysis and the results of applying the methods of analysis, followed by a discussion (Section 5). Finally, Section 6 concludes the paper and outlines our future work.

2 Related Work

2.1 Regression Models

Regression techniques [16] are a kind of algorithmic techniques which looks for an equational model to fit a set of observed data values.

Linear Regression (LR) is the classical linear regression model. It is assumed that there is a linear relationship between a dependant variable (e.g., effort) with a set of or independent variables, i.e., attributes (e.g. *size in function points, team size, development platform,* etc.). The aim is to adjust the data to a model so that $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + + \beta_k x_k + e$.

The linear least square method finds the line that minimises the sum of squared errors. A problem with this method is that it assumes a normal distribution and cannot cope well with outliers. Regression analysis can be made more robust for outliers using the least median squares method. Least median square regression analysis is suitable for small data-sets, and it is sensitive to abnormal observations and errors. Least Median Square (LMS) is a robust regression technique that includes outlier detection.

2.2 Rules, Decision Trees and Model Trees

A *rule-based system* [9] consists of a library of rules of the form: *if* (assertion) *then* action. Such rules are used to elicit information or to take appropriate actions when specific knowledge becomes available. These rules reflect a way to reason about the relationships within the domain. Their main advantage is

their simplicity. However, they are mainly appropriate for deterministic problems, which is not usually the case in software engineering [1]. To overcome this problem, rules can also contain a certainty measure in the premises and/or in the conclusions.

Decision trees [3] are used for predicting or explaining outputs from observations. In such a tree, each node is a leaf indicating a class or an internal decision node that specifies some test to be carried out. If the output values conform to intervals, then the decision trees are called regression trees, whereas if they do correspond to a nominal or ordinal scale they are called classification trees. There are many tree-building algorithms such as C4.5 (Quinlan, 1993) which determine which attributes best classifies the remaining data, and then the tree is constructed iteratively. The main advantage of decision trees is their immediate conversion to rules that can be easily interpreted by decision-makers. For numeric prediction in data mining, it is common to use regression trees or model trees [3]. Both techniques build a decision tree structure where each leaf is responsible for a particular local regression of the input space, in our case the software engineering dataset. The difference between them is that while a regression tree generates constant output values for subsets of input data (zero-order models), model trees generate linear (first-order) models for each subset.

The *M5 algorithm* builds trees whose leaves are associated to multivariate linear models and the nodes of the tree are chosen over the attribute that maximizes the expected error reduction as a function of the standard deviation of output parameter. In this paper, we have applied a version of M5, called M5P -Prime- implemented in the WEKA toolkit [21] to the ISBSG dataset [10]. Weka's M5 algorithm actually builds a decision tree which divides the attribute space in an orthohedric clusters, with the border parallel to the axis. An advantage of model trees is that they can be easily converted into rules; each branch of the tree has a condition as follows: $attribute \leq value$ or $attribute > value$.

We will now explain how the M5 algorithm works using the *Reality* dataset, which is a subset of the ISBSG repository with 709 projects used by the Reality estimation tool accompanying the ISBSG repository and further explained in the next section. For example, Weka's M5 algorithm created only a decision node to calculate the *NormalisedWorkEffort*, and therefore, two linear models, LM1 and LM2:

```
UnadjustedFunctionPoints <= 343:
LM1 (510/53.022\%)
NormalisedWorkEffort =
 90.5723 * DevelopmentPlatform=MF,MR
 + 63.5148 * LanguageType=ApG,3GL,2GL
 + 628.9547 * LanguageType=3GL,2GL
 + 184.9949 * ProjectElapsedTime
 + 10.9211 * UnadjustedFunctionPoints
 - 545.8004
```

```
UnadjustedFunctionPoints > 343:
LM2(199/318.225\%) NormalisedWorkEffort =
  10189.7332 * DevelopmentPlatform=MF,MR
  - 5681.5476 * DevelopmentPlatform=MR
  + 155.8191 * LanguageType=ApG,3GL,2GL
  + 5965.379 * LanguageType=3GL,2GL
  + 551.4804 * ProjectElapsedTime
  + 4.3129 * UnadjustedFunctionPoints
  - 8118.3275
```

The two branches generated are interpreted as follows: if *UnadjustedFunctionPoints*, i.e. size of the system, is less than 343 then we apply LM1 otherwise LM2. In this case, both LM1 and LM2 are lineal models where the *Normalised-*

WorkEffort is the dependent variable; M5 however can assign to the dependent variable either a constant or a linear equation (in the majority of the cases).

The categorical data of the linear regression function obtained by Weka is calculated by substituting the value for the appropriate value wherever it occurs. For example, if we had an instance with *DevelopmentPlatform* equals to MF, *LanguageType* equals to *ApG* and *UnadjustedFunctionPoints* less than 343.

For evaluating each categorical expression, if the value of the category on the left hand side is equal to any of the categories on the right hand side of the equation, then we substitute the entire equation with value 1; otherwise with the value 0. Following the example we obtain:

```
LM num: 1 NormalisedWorkEffort =
    90.5723 * 1, MR
    + 63.5148 * 1
    + 628.9547 * 0
    + 184.9949 * ProjectElapsedTime
    + 10.9211 * UnadjustedFunctionPoints
    - 545.8004
```

Further information provided by the M5 algorithm within brackets is as interpreted as follows. For the LM1 branch, there are 510 instances and the error in that leave is 53.022%; for LM2, there were 199 projects and the error was 318.225%.

3 Evaluation of the Techniques

We have created 2 subsets of the ISGBN repository. One of them, called the Reality dataset, is included in the repository as part of the ISBSG Reality Checker tool (used for effort estimation). Datasets are divided into training and test, such that approximately 2/3 of the instances are used for training and the other 1/3 of the instances are used for testing, i.e. evaluation. This is a common practice in data mining; other other techniques such as cross validation can be more accurate, they are however more complex. In statistics as well as in data mining, with linear regression models used in this work, the goodness of fit of a model is usually measured by the correlation and by the *mean squared error*. In the software Engineering domain however, it is common to compare the goodness of each technique using the Mean Magnitude of Relative Error (MMRE) and Pred(%), proposed by Conte et al. [4]:

- MMRE is calculated as $\frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{e_i - a_i}{e_i} \right|$, where n is the sample size , e_i is the estimated value for the *i-th* element and a_i is the actual value.
- Prediction at Level l –$Pred(l\%)$– is defined as the number of cases whose estimations are under the $l\%$, divided by the total number of cases. For example, $Pred(25) = 0.75$ means that 75% of cases estimates are within the inside 25% of its actual value.

In Software Engineering, standard criteria for a model to acceptable are roughly $Pred(25) \geq 0.75$ and $MMRE \leq 0.25$ [6].

4 Approach and Results

The International Software Benchmarking Standards Group (ISBSG), a non-profit organization, maintains a software project management repository from a variety of organizations. The ISBSG checks the validity and provides benchmarking information to companies submitting data to the repository. Furthermore, it seems that the data is collected form large and successful organizations. In general, such organizations have mature processes and well established data collection procedures. In this work, we have used release 8, which contains 2028 projects and more than 55 attributes per project.

Before using the dataset, there are a number of issues to be taken into consideration. An important attribute is the quality rating given by the ISBSG can be from A (where the submission satisfies all criteria for seemingly sound data) to D (where the data has some fundamental shortcomings). According to ISBSG only projects classified as A or B should be used for statistical analysis. Also, many attributes in ISGSB are categorical attributes or multi-class attributes that need to be pre-processed for this work (e.g. the project scope attribute which indicates what task were included in the project work effort –planning , specification, design, development and testing– were grouped). Another problem of some attributes is the large number of missing instances. As a result, we had to do some pre-processing. We selected some attributes and instances manually. There are quite a large number of variables in the original dataset that we did not consider relevant or they had too many missing values to be considered in the data mining process. From the original database, we only considered the IFPUG estimation technique and those that can be considered very close variations of IFPUG such as NESMA [12] or Dreger [5] .

In our study, we have selected *NormalisedWorkEffort* or *SummaryWorkEffort* as dependent variable provided by the ISBGN dataset . The normalized work effort is an estimate of the effort for the whole software life-cycle even if the project did not cover all the phases in the software development life-cycle. Summary work effort is the actual effort even if the project did not carry out the whole life-cycle. Both values are the same for projects covering the whole life-cycle or projects where it is not known if they covered the whole life-cycle. For each dataset, we divided the dataset into a training and test using Weka utilities to creates stratified cross-validation folds [7]. This means that per default class distributions are approximately retained within each fold. The training dataset contains approximately 2/3 of instances and the remaining 1/3 of instances was used for validation.

We compared the outputs using algorithms provided by Weka's M5 models, Multivariate Linear Regression (MLR), or just Linear Regression (LR) for simplicity, and Least Median Squares (LMS) for each dataset.

DS1. The Reality dataset if composed of 709 instances and 6 attributes (*DevelopmentType, DevelopmentPlatform, LanguageType, ProjectElapsedTime, NormalisedWorkEffort, UnadjustedFunctionPoints*). The dependant variable that we used for this dataset is the *NormalisedWorkEffort*. Table 1 compares the goodness of the results for the reality dataset.

Table 1. DS1. Reality dataset results

	M5	***LeastMedSq***	***LR***
Correlation coefficient	0.36	0.06	0.37
Mean absolute error	4829.08	5817.38	5244.29
Root mean squared error	15715.46	18583.45	15612.01
Relative absolute error	74.18%	89.36%	80.56%
Root relative squared error	93.54%	110.6143%	92.92%
MMRE	1.99	1.44	2.62
Pred(25)	0.20	0.17	0.13
Pred(30)	0.24	0.22	0.16

Table 2. DS2. Dataset results

	M5	***LeastMedSq***	***LR***
Correlation coefficient	0.87	0.8108	0.78
Mean absolute error	1191.08	3497.23	3340.26
Root mean squared error	7978.20	12437.35	9482.81
Relative absolute error	20.32%	59.67 %	56.99%
Root relative squared error	54.07%	84.30%	64.27%
MMRE	0.39	0.76	1.69
Pred(25)	0.70	0.29	0.23
Pred(30)	0.73	0.36	0.28

DS2. The dataset DS2 is composed of 1390 instances and 15 attributes (*FP, VAF, MaxTeamSize, DevType, DevPlatf, LangType, DBMUsed, MethodUsed, ProjElapTime, ProjInactiveTime, PackageCustomisation, RatioWEProNonPro, TotalDefectsDelivered, NormWorkEff, NormPDR*). The dependant variable for this dataset is the *NormalisedWorkEffort*.

5 Discussion

The M5 models provide some advantages from both quantitative and qualitative point of view when compared with simpler regression methods. From a quatitative point of view, we can conclude that the goodness, i.e., estimation accuracy, of M5 is better than the classical linear regression or least mean squares (cf. Tables 1 and 2). The M5 algorithm greatly improved estimates to levels that could be considered as acceptable by the software engineering community. Also, when compared with other techniques, the M5 algorithm is able to handle both continuous and categorical variables.

In addition to greater accuracy, the M5 model provides other advantages from a qualitative point of view. Firstly, each subset is clearly defined in the sense that new instances are easily assigned to a local model. The second benefit is that decision trees are easily understandable by users in general and by project managers in particular as we can read them as rules. Each branch of the tree has a condition as follows: $attribute \leq value$ or $attribute > value$. Such conditions are

frequently used by experts in all sciences. Also, this provides a clear indication of which variables are most important for prediction (as conditionals in the branches of the tree). The leaves of the tree allow project managers to gain further knowledge into the characteristics of the dataset.

6 Conclusions

This paper presented the results of applying the M5 algorithm as a estimation technique to 2 datasets generated from the ISGSB repository. The M5 algorithm assigns to each of the generated local model a linear regression formula. We also applied the classical linear regression and least median squares and M5 for effort estimation. When we compared the goodness of such methods, results show that there is an improvement in the accuracy of the results when using such local models. Furthermore, M5 can help us to understand the datasets better; the tree (rules) generated with M5 provides project managers with a better understanding of what attributes are more important in a particular dataset. Finally, we believe that M5 in particular and clustering techniques in general, provide some advantages to project managers while keeping the estimation process within reasonable complexity.

Further work will consist of using data mining techniques for characterizing not only the instances but also the attributes (in this work, the attributes were selected manually using expert knowledge). More needs to be done also understanding and comparing different clustering techniques to create segmented models and its usefulness for project managers.

Acknowledgments

The research was supported by the University of Reading and the Spanish Research Agency (CICYT TIN 2004-06689-C03 – The INGESOFT Project).

References

1. Aguilar–Ruiz J.S., Riquelme J.C., Ramos I. and Toro M.: An evolutionary approach to estimating software development projects. *Information and Software Technology*, 14(43):875–882, 2001
2. Boehm, B. 1981, Software Engineering Economics, Prentice-Hall, 1981
3. Breiman, L., Friedman, J., Olshen, R. and Stone, C. J.,*Classification and Regression Trees*, Chapman and Hall, New York, 1984.
4. Conte SD, Dunsmore HE and Shen V. 1986. *Software Engineering Metrics and Models*, Benjamin/Cummings.
5. Dreger, J. 1989, Function Point Analysis, Englewood Cliffs, NJ, Prentice Hall,
6. Dolado, J.J.: On the problem of the software cost function. *Information and Software Technology, 43:61–72, 2001.*
7. *Fayyad, U.M. and Irani, K.B.: Multi-interval discretisation of continuous valued attributes for classification learning. In* Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, 1993.

8. Finnie, G.R., Wittig, G.E., and Desharnais, J.-M.: A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. *Journal of Systems and Software*, 39(3):281–289, 2000.
9. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11:63–91, 1993.
10. ISBSG, 2004. International Software Benchmarking Standards Group (ISBSG), Web site: http://www.isbsg.org/
11. Mitchell, T.: *Machine Learning*. McGraw Hill, 1997.
12. NESMA, 1996. NESMA FPA. Counting Practice Manual Version 2.0.
13. PRICE, 2005. Price S. Web Site: http://www.pricesystems.com/
14. Quinlan JR. 1992. Learning with continuous class. *Proc. of the 5th Australian Joint Conference on Artificial Intelligence*: 343-348, World Scientific.
15. Quinlan, J.R.: *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, California, 1993.
16. Rousseeuw, P. J. and Annick M. L., *Robust Regression and Outlier Detection*, New York: John Wiley & Sons, 1987.
17. Shepperd, M. and Schofield, C.: Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12):736–743, 2000.
18. Srinivasan, K. and Fisher, D.: Machine Learning Approaches to Estimating Software Development Effort. *IEEE Transactions on Software Engineering*, 21(2): 126–137, 1995.
19. Walkerden, F. and Jeffery, R.: An empirical study of analogy-based software effort estimation. *Empirical Software Engineering*, 42:135–158, 1999.
20. Wang, Y. and Witten, I.H.: Induction of model trees for predicting continuous classes. *Proceedings of the poster papers of the European Conference on Machine Learning*. University of Economics, Faculty of Informatics and Statistics, Prague.
21. Witten I. and Frank E. 1999. *Data Mining Practical: Machine Learning Tools and techniques with Java implementations.* Morgan Kaufmann