Selecting the Appropriate Machine Learning Techniques for the Prediction of Software Development Costs

Stamatia Bibi, Ioannis Stamelos
Aristotle University of Thessaloniki, Department of Informatics, 54124, Greece, {sbibi,stamelos}@csd.auth.gr
WWW home page: http://sweng.csd.auth.gr

Abstract. This paper suggests several estimation guidelines for the choice of a suitable machine learning technique for software development effort estimation. Initially, the paper presents a review of relevant published studies, pointing out pros and cons of specific machine learning methods. The techniques considered are Association Rules, Classification and Regression Trees, Bayesian Belief Networks, Neural Networks and Clustering, and they are compared in terms of accuracy, comprehensibility, applicability, causality and sensitivity. Finally the study proposes guidelines for choosing the appropriate technique, based on the size of the training data and the desirable features of the extracted estimation model.

1 Introduction

Cost estimation refers to the prediction of the human effort (typically measured in man-months) and time needed to develop a software artifact [5]. The estimation is based on various attributes of the software project, such as language type, personnel skills, computer platform, project constraints, etc. Such information is found in historical cost data bases that may be small data sets coming from the estimating software organization or large multi-organizational data bases. Although commercial cost estimation models are available, ad hoc models, based on selected past project data are considered the best approach.

Many studies have been published so far regarding the applicability of machine learning approaches to software cost estimation. Due to the different parameters of the experiments, the different data sets used and the varying evaluation methods no general conclusions can be directly drawn concerning the suitability of each method. Target of this study is to examine the compiled knowledge and experience coming from the application of machine learning techniques. The final result is the extraction

Please use the following format when citing this chapter:
Bibi, Stamatia, Stamelos, Ioannis, 2006, in IFIP International Federation forInformation Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 533–540

of a set of guidelines that will help an estimator select the appropriate method based on the environment in the context of which the estimation is performed.

In the following, the basic criteria for judging a cost estimation technique are presented as research questions:

How **comprehensible** is the extracted model? This is important in a problem domain such as project effort prediction since the estimator must trust the model's output, otherwise the prediction may be rejected by management.

How **applicable** is the method? The cost, effort and time needed to learn and extract an estimation model may be crucial for estimation environments that lack human experts, time or money to devote to such activities. It is important to possess effective tools that will support method application.

Does the model address **causality**? When a software project is less productive compared to others it is important to have a clue of the main reasons that caused this. Estimation should include causality, in a way that it is easily interpreted. A simple cost estimate is not enough any more, it should be accompanied with supporting evidence justifying the reasons for this estimate.

Does the method handle **missing values**? In software engineering data sets noise is a usual phenomenon. If a method can ignore missing values without excluding projects or independent variables then it has an advantage compared to a method that can handle only complete data. This is an important feature for software data sets that are usually small and incomplete.

Is **uncertainty** considered in the models? In every estimation process, uncertainty is an inevitable element that should be well managed. This element is even more important in software cost estimation models, where money, time, resources and customer relationships depend on this estimation.

What is the **accuracy** of the models? The accuracy of an estimation model is among the most important features of a method. Due to the fact that each study has different parameters direct comparison of the accuracy of the models cannot be performed. Though, several conclusions can be drawn about the situations under which each method outperforms the rest.

Does the method support a **dynamic approach** to the problem? For estimation problems that evolve over time, such as software cost estimation, techniques that can deal with updated information have an advantage over static techniques.

What is the **sensitivity** of the method? The sensitivity of a method involves its ability to produce accurate estimations even when several parameters change.

The paper is structured as follows: Section 2 shows how machine learning techniques are applied for estimating software development cost. Section 3 reports the results of published research studies and compares techniques. Section 4 compiles estimation guidelines for determining the appropriate technique(s) according to the data in hand. Section 5 concludes the paper and provides research directions.

2 Machine Learning Techniques in Software Cost Estimation

In this section we will demonstrate how the five machine learning methods can be used for softeware cost estimation.

An AR [1] coming from the domain of software cost estimation will have as Rule Body certain software project attribute values and as a Rule Head a productivity (or cost, or effort) value. A simple example of an AR is presented in table 1.

Table 1. Association Rule for Software Productivity Estimation

If language used = cobol and development type= enhancement	
then 40<productivity ≤60	support=5% confidence= 85%

This rule is interpreted as following: If the language that will be used for the development of new project is COBOL and the development type of the project is enhancement then there is 85% (confidence value) probability that the productivity value of the project will be between 40 and 60 lines of code per hour. This rule is classifies correctly 5% of the instances in the training data set.

Bayesian Belief Networks [13] in software cost estimation are directed acyclic graphs with each node representing a software project variable, or software development effort. A simple Bayes Network estimating software effort is the one presented in figure 1.

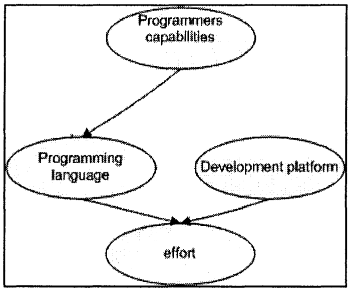

Fig. 1. A BBN for software effort estimation

Attached to the node of effort there is a node probability table that provides possible values of the effort based on the combination of values that the programming language and the development platform nodes take.

CART is a widely used statistical procedure for producing classification and regression models with a tree-based structure in predictive modeling [2]. The CART tree model of figure 2 for software cost estimation consists of an hierarchy of univariate binary decisions that leads to the prediction of software productivity.

Clustering [9] is the process of decomposing or partitioning a data set into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups. In software cost estimation to our knowledge clustering have been used mainly for selecting similar groups of project in which another estimation technique will be used to provide a prediction.

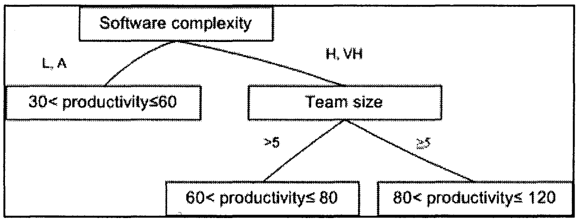

Fig. 2. CART for software productivity estimation

NNs [11] are massively parallel systems comprising simple interconnected units, artificial neurons. The neuron computes a weighted sum of its inputs and generates an output if the sum exceeds a certain threshold. This output then becomes an excitatory or inhibitory input to other neurons in the network. The process continues until one or more outputs are generated. In software cost estimation the inputs are the project attribute values, the output is the estimation of productivity.

3 Machine learning in software cost estimation

In this section we review the results obtained by various researchers and provide a comparative table based on the model quality criteria we proposed in the Introduction.

Rule induction has been the target of several studies [1], [2], [6], [11], [15]. The studies are differentiated by the algorithm used to extract rules. Studies [1], [2] extract association rules directly from the learning data set. Study [11] extracts mutually exclusive rules that can form regression trees, while studies [6] and [15] utilize rules as a support to fuzzy models. The accuracy of the method is a central theme to almost all studies apart from [11].

The studies considered identify comprehensibility of the results as a clear advantage of the method. Rules are among the most representative forms of human notion, they are transparent and therefore easily read and understood. Rule representation style helps the estimator understand the prediction and any underlying assumptions upon which it is based. Rules may be rephrased and provided to offer a clearer explanation as to how a prediction has been made and the evidence on which the prediction is based. Additionally, rules have the ability to include uncertainty in the prediction as each rule is accompanied by two probabilities that show the statistical validity and strength of the rule in the learning data set.

The accuracy of AR as an estimation method depends mainly on the number of rules extracted and whether AR has support from other methods. For example in study [11] when the pruned set of rules is evaluated the accuracy is decreased. In studies [1], [2] the number of rules extracted is large and therefore the estimation accuracy increased. The main problem of the method is that when rule induction is used, estimation accuracy is relatively low. On the other hand when association rules are used accuracy is higher but it cannot be guaranteed that the model will be able to classify all new projects. Another problem of the method is its sensitivity to parameter selection (number of rules, independent variables participating in the model, size of the training set). Changing one of the parameters may cause large model output fluctuation and reduce the accuracy of the estimates.

Bayesian Belief Networks is a relatively new approach to the problem of software cost estimation. Direct application of BBN on software cost estimation is found in two studies [3], [13], while [4] utilized Bayesian analysis for calibrating the well-known COCOMO II model. Bayesian analysis is a well-defined and rigorous process of inductive reasoning. A distinctive feature of the Bayesian approach is that it permits the investigator to use both sample (data) and prior (expert-judgment) information in a logically consistent manner in making inferences [4]. This is done by using Bayes' theorem to produce a `postdata' or posterior distribution for the model parameters. Using Bayes' theorem, prior (or initial) values are transformed to postdata views.

Among the other advantages of the method is its ability to represent domains that evolve over time. BBN can offer a dynamic approach to the problem of software cost estimation as they are able to update their estimates when more information is included in the model. In general, using BBN estimation uncertainty is captured elegantly in the node probability tables that accompany each model and causality is addressed by pointing out the variables that mainly affect cost. Among the disadvantages of the method is that in small data sets BBN can be inaccurate, when particular combination of values of the independent variables are excluded from the learning set. Also the method cannot handle missing values, and though BBN support visual representation of the results the estimator has to be knowledgeable of the method in order to interpret the quantitative results as well.

Classification and Regression Trees are explored and applied in the comparative studies [2], [4], [7], [8]. In these studies the accuracy of the method usually is average to low and it is a fact that CART has never achieved the best performance among the compared methods. The main advantage of the method is that it is easily applied and produces comprehensible models. Additionally CART has the ability to classify all potential projects to a cost value even if the attributes of the project under estimation have not appeared in the training data set. CART can also handle missing values but the performance of the method increases when there is no noise in the learning data. Among the problems of the method is that the splitting of the data in each node sometimes seam unreasonable, as the method cannot deal with scale variables. Also due to the automated pruning of the tree that CART tools support, in order to avoid over-fitting to the data, several cost values that are loosely represented in the data tend to be omitted, resulting in the misclassification of projects belonging to minor cost intervals.

Neural networks have been often applied and compared in the domain of software cost estimation [6], [9], [11], [12], [14]. Most studies concerned with the use of NNs to predict software development effort have focused on comparing their accuracy with that of algorithmic models, rather than on the suitability of the approach for building software effort prediction models. In most studies where NN have been utilized the technique has relatively high accuracy and usually outperforms the rest of the techniques. We note, however, that other factors such as explanatory value and configurability are poorly addressed by the method. NN produce results in the form of a "black box" and therefore it is difficult for an estimator to understand the rationale under which the prediction has been made. Also the application of NN is a difficult, time consuming procedure. Among the main drawbacks of the method is its sensitivity to the data. Overfitting of the models to the training data is another reported problem.

Clustering has been applied on software cost estimation in combination with other methods [9]. First data are split into homogeneous clusters and then another method is applied. Clustering provides important information when the initial estimation models from the training data set are not satisfying and the models have to be re-generated using a part of the data. Clustering is useful for selecting data that will be in the same set for extracting estimation models.

Table 2 summarizes the results of the examined studies and provides a tool for comparing the various machine learning techniques.

Table 2. Comparison of machine learning methods in terms of quality criteria

Methods/ Criteria	**AR**	**BBN**	**CART**	**CLUSTERING**	**NN**
comprehensibility	high	medium	high	medium	low
applicability	medium	medium	high	high	low
causality	high	high	medium	medium	low
missing values	yes	no	yes	yes	no
uncertainty	medium	high	medium	medium	low
accuracy	depends	medium	low	NA	high
dynamic update	NA	high	NA	NA	high
sensitivity	high	medium	Medium	low	high

4 Choosing the appropriate technique

A standard approach in practical software cost estimation is to apply more than one technique [10], in order to avoid biased estimates. Based on the above assumptions we suggest a decision tree for selecting the appropriate technique according to the size of the data set and the importance that the estimator assigns to each model criterion. The overall approach is presented in the form of a decision tree presented in figure 3.

When the number of training data is relatively large (e.g. a large multi-organization data base or a rich local cost data base is used) the suggested method is BBN. BBN have the ability to represent various aspects of the cost estimation problem and when applied to large data sets potential disadvantages of the method are handled. In case the estimator is interested only in high accuracy of the estimates, and has the available resources to apply the method then the use of NN is suggested.

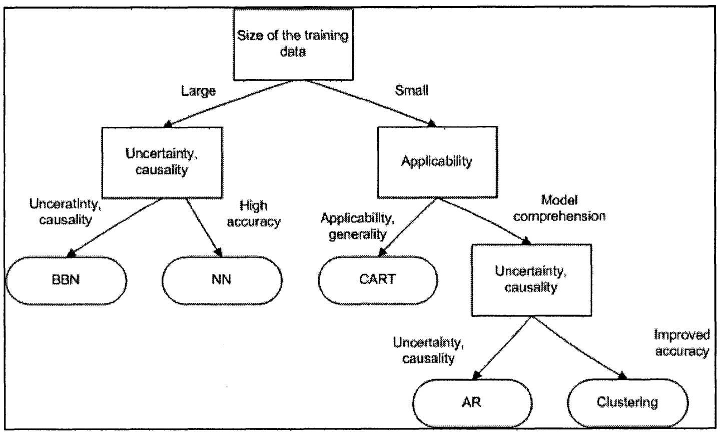

Fig. 3. Decision tree for selecting software cost estimation method

In small data sets (typically local software cost data bases) the selection of the method varies based on the situation of the estimation. When there is little time or the available tools and the estimation model must classify all possible projects then the use of CART is suggested, although accuracy may be a problem. On the other hand if uncertainty and causality of the model play important role in managerial decisions such as selection of tools, staff and platform for the project, the use of AR is proposed. AR will show which attributes affect the effort of a software project and the level they influence it. Finally if none of the previous hypotheses is valid and the models extracted directly from one machine learning technique are not satisfactory, the use of clustering is suggested along with re-application of the first technique.

5 Conclusions and future work

In this paper we examined five machine learning techniques in terms of accuracy, comprehensibility, causality, applicability, sensitivity, uncertainty, handling of missing values and dynamic update. Several advantages and disadvantages of AR, BBN, CART, clustering and NN have been indicated. Based on the assumptions coming from current literature regarding machine learning techniques, we suggested a number of guidelines for selecting the appropriate estimation methods based on the needs and the resources of the estimator.

It is obvious that for more detailed analysis of the above issues further research has to be done applying and comparing directly the above methods under the same estimation situation. Future work may also involve the application of the methods in large data sets, with missing values and noise. Also possible combination of machine learning methods for software cost estimation is definitely an interesting approach for solving the problems each method has alone.

References

1. S. Bibi, I. Stamelos, L. Angelis: Software Productivity estimation based on Association Rules, In Proc. of 11th European Software Process Improvement Conference, pp. 13A1, Trondheim, Norway, November 2004.
2. S. Bibi, I. Stamelos, L. Angelis: Software Cost Prediction with Predefined Interval Estimates, In Proc. of 1st Software Measurement European Forum, pp. 237-246, Rome, Italy, January 2004.
3. S. Bibi, I. Stamelos, L. Angelis: Bayesian Belief Networks as a Software Productivity Estimation Tool, In Proc. of 1st Balkan Conference in Informatics, pp. 585-596, Thessaloniki, Greece, November 2003.
4. L.C. Briand, K.E. Emam, D. Surmann, I. Wieczorek, K. Maxwell, An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques, In Proc. of International Conference on Software Enginnering, pp. 313-322, CA, USA, May 1999
5. S. Chulani, B. Boehm, B. Steece, Bayesian Analysis of Empirical Software Engineering Cost Models, IEEE Transactions on Software Engineering, 25(4), pp. 573-583, 1999.
6. X. Huang, L. Capretz, J. Ren, D. Ho, A Neuro-Fuzzy Model for Software Cost Estimation, In Proc. of 3d International Conference on Quality Software, pp. 126- 133, Texas, USA, November, 2003.
7. R. Jeffery, M. Ruhe, I. Wieczorek, Using Public Domain Metrics to Estimate Software Development Effort, In Proc. of International Software Metrics Symposium, pp. 16-27, California, USA, April, 2001.
8. B. Kitchenham, A procedure for analyzing unbalanced datasets, IEEE Transactions on Software Engineering, 24(4), pp. 278-301, 1998.
9. A. Lee, C. Cheng, J. Blakrishnan, Software development cost estimation: Integrating neural network with cluster analysis, Information and Management, 34 (1), pp. 1-9, 1998.
10. S. MacDonell, M. Shepperd, Combining techniques to optimize effort predictions in software project management, Journal of Systems and Software , 66 (2), pp. 91-98, 2003.
11. C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, S. Webster, An Investigation of Machine Learning Based Prediction Systems, Journal of Systems and Software, 53, pp. 23-29, 2000.
12. K. Srinivasan, D. Fisher, Machine Learning Approaches to Estimating Software Development Effort, IEEE Transactions on Software Engineering, 21(2), pp. 126-137, 1995.
13. I. Stamelos, P. Dimou, L. Angelis, E. Sakellaris, On the Use of Bayesian Belief Networks for the Prediction of Software Development Productivity, Information & Software Technology, 45, pp. 51-60, 2003.
14. N. Tadayon, Neural Network Approach for Software Cost Estimation, In Proc of International Conference on Information Technology, pp. 815-818, 2, LA, USA, April, 2005.
15. Z. Xu, T. Khoshgoftaar: Identification of fuzzy models of software cost estimation, Fuzzy Sets and Systems, pp. 141-163, 145(1), 2004.