Reliable Confidence Intervals for Software Effort Estimation

Harris Papadopoulos, Efi Papatheocharous and Andreas S. Andreou

Abstract This paper deals with the problem of software effort estimation through the use of a new machine learning technique for producing reliable confidence measures in predictions. More specifically, we propose the use of Conformal Predictors (CPs), a novel type of prediction algorithms, as a means for providing effort estimations for software projects in the form of predictive intervals according to a specified confidence level. Our approach is based on the well-known Ridge Regression technique, but instead of the simple effort estimates produced by the original method, it produces predictive intervals that satisfy a given confidence level. The results obtained using the proposed algorithm on the COCOMO, Desharnais and ISBSG datasets suggest a quite successful performance obtaining reliable predictive intervals which are narrow enough to be useful in practice.

1 Introduction

Accurate software cost estimation has always been a challenging subject impelling intensive research from the software engineering community for many years now [11]. Especially over the last 50 years researchers have been working to improve the estimation accuracy of the methods proposed and provide

Harris Papadopoulos
Computer Science and Engineering Department, Frederick University, 7 Y. Frederickou St., Palouriotisa, Nicosia 1036, Cyprus. e-mail: H.Papadopoulos@frederick.ac.cy

Efi Papatheocharous
Department of Computer Science, University of Cyprus, 75 Kallipoleos str., CY1678 Nicosia, Cyprus e-mail: efi.papatheocharous@cs.ucy.ac.cy

Andreas S. Andreou
Department of Computer Science, University of Cyprus, 75 Kallipoleos str., CY1678 Nicosia, Cyprus e-mail: aandreou@cs.ucy.ac.cy

a competitive edge for software companies and managers. An accurate estimate, especially from the early stages of the software project life-cycle, could provide more efficient management over the whole software project resources and processes. Such estimates, even for well-planned projects, are hard to make and thus it may prove helpful to handle the uncertainty of the estimates through an effort prediction interval. Nevertheless, the difficulty to produce intervals for effort estimation reflecting a new project remains due to the high level of complexity and uniqueness of the software process. Estimating robust confidence intervals of the required software costs, as well as selecting and assessing the most suitable cost drivers for describing the escalating behavior of effort, both remain difficult issues to tackle and are constantly at the forefront right from the initiation of a project and until the system is delivered.

In this work, we aspire to tackle the uncertainty of the software process and the volatility of leading cost factors in the development environment by producing confidence intervals for software effort. Most cost models and techniques proposed in literature provide a single estimate for effort and usually have poor generalization ability [8]. Our approach attempts to promote reliable confidence intervals to reach better solutions in such approximation problems so as to alleviate the deficiencies of the techniques proposed so far and to address the problem in a possibly more effective and practical manner.

The present paper proposes the use of a novel machine learning technique, called Conformal Prediction (CP) [24], which can be used to produce predictive intervals, or regions, that satisfy a required level of confidence. Conformal Predictors (CPs) are based on conventional machine learning algorithms and transform the output of these algorithms from point to confidence interval predictions. The most important property of CPs is that they are well calibrated, meaning that in the long run the predictive intervals they produce for some confidence level $1-\delta$ will not contain the true label of an example with a relative frequency of at most δ [16]. Furthermore, this is achieved without assuming anything more than that the data are distributed independently by the same probability distribution (i.i.d.), which is the typical assumption of most of the machine learning methods. In this paper we use the Ridge Regression Conformal Predictor (RRCP), which, as its name suggests, is based on the well known Ridge Regression (RR) algorithm. We applied RRCP to three empirical cost datasets, namely the COCOMO, the Desharnais and the ISBSG, each including a set of different cost factors measured over a series of completed software projects in the past.

The rest of this paper is organised as follows: Section 2 presents a brief literature overview of data driven cost estimation and outlines similar machine learning attempts reported in literature. Section 3 presents the background theory behind RR and CP, while section 4 provides a description of the experiments and associated results produced using the aforementioned datasets. Finally, section 5 summarises the findings of the paper and suggests future research steps.

2 Brief Literature Overview

Recently, many machine learning approaches have been investigated in the field of software effort estimation, such as neural networks [20], neuro-fuzzy approaches [9], support vector regression [17], genetic algorithms [1] and rule induction algorithms [14] with quite satisfactory results. Machine learning techniques use empirical data from past projects to build a model that can then be employed to predict the effort of a new project. Although such data-driven cost estimation methods employed in empirical software engineering studies report quite encouraging results, all these techniques suffer from lack of uncertainty value consideration. Most existing methods for software effort estimation only provide a single value as their estimation for the effort of a given project, without any associated information about how "good" this estimation may be. Yet, the provision of an interval which will include the effort of a project at a given level of confidence would have been much more informative [8]. For this reason, recent studies such as [2, 8, 12, 13], employed different techniques for producing predictive intervals.

A relatively new development in the area of Machine Learning is the introduction of a novel technique, called Conformal Prediction [24], that can be used for complementing the predictions of traditional algorithms with provably valid measures of confidence. Therefore, this technique is ideal for dealing with the problem of confidence interval prediction for software cost estimation. What we call in this paper CP was first proposed in [6] and then greatly improved in [23]. In both [6] and [23] the base algorithm used was support vector machine. Slightly later CP was applied to other algorithms such as Ridge Regression [15] and k-nearest neighbours for both classification [21] and regression [19]. At the same time, in an effort to improve the computational efficiency of CPs, a modification of the original approach was developed, called Inductive Conformal Prediction [18]. Since then CP has been applied successfully to problems such as the early detection of ovarian cancer [7] and the classification of leukaemia subtypes [3].

3 Ridge Regression Conformal Predictor

We first briefly describe the Conformal Prediction (CP) framework and then focus on its application to the dual form Ridge Regression (RR) algorithm [22]. RR is an improvement of the classical Least Squares technique and its one of the most widely used regression algorithms. The main reason we chose the dual form RR method is that it uses kernel functions to allow the construction of non-linear regressions without having to carry out expensive computations in a high dimensional feature space.

We are interested in making a prediction for the label of an example x_l, based on a set of training examples $\{(x_1, y_1), \ldots, (x_{l-1}, y_{l-1})\}$; where each

$x_i \in \mathbb{R}^d$ is the vector of attributes for example i and $y_i \in \mathbb{R}$ is the label of that example. Our only assumption is that all (x_i, y_i), $i = 1, 2, \ldots$, are i.i.d. CP considers every possible label $\tilde{y}$ of the new example x_l and assigns a value α_i to each pair (x_i, y_i) of the extended set

$$\{(x_1, y_1), \ldots, (x_l, \tilde{y})\} \tag{1}$$

that indicates how strange, or non-conforming, that pair is for the rest of the examples in the same set. This value, called the *non-conformity score* of the pair (x_i, y_i), is calculated using a traditional machine learning algorithm, called the *underlying algorithm* of the corresponding CP. More specifically, the non-conformity score of a pair (x_i, y_i) is the degree of disagreement between the actual label y_i and the prediction $\hat{y}_i$ of the underlying algorithm, after being trained on (1). The function used for measuring this degree of disagreement is called the *non-conformity measure* of the CP.

The non-conformity score α_l is then compared to the non-conformity scores of all other examples to find out how unusual $(x_l, \tilde{y})$ is according to the non-conformity measure used. This is achieved with the function

$$p((x_1, y_1), \ldots, (x_l, \tilde{y})) = \frac{\#\{i = 1, \ldots, l : \alpha_i \geq \alpha_l\}}{l}, \tag{2}$$

the output of which is called the p-value of $\tilde{y}$, also denoted as $p(\tilde{y})$, since this is the only unknown value in (1). An important property of (2) is that $\forall \delta \in [0, 1]$ and for all probability distributions P on Z,

$$P^l\{((x_1, y_1), \ldots, (x_l, y_l)) : p(y_l) \leq \delta\} \leq \delta; \tag{3}$$

a proof can be found in [16]. This makes it a valid test of randomness with respect to the i.i.d. model. According to this property, if $p(\tilde{y})$ is under some very low threshold, say 0.05, this means that $\tilde{y}$ is highly unlikely as the probability of such an event is at most 5% if (1) is i.i.d. Assuming it were possible to calculate the p-value of every possible label following the above procedure, then we could exclude all labels with a p-value under some very low threshold, or *significance level*, δ and have at most δ chance of being wrong. Consequently, given a confidence level $1 - \delta$ a regression CP outputs the set

$$\{\tilde{y} : p(\tilde{y}) > \delta\}, \tag{4}$$

in other words the set of all labels that have a p-value greater than δ. Of course it is impossible to explicitly consider every possible label $\tilde{y} \in \mathbb{R}$, so the RRCP follows a different approach which allows it to compute (4) efficiently. This approach, proposed in [15], is described in the next few paragraphs.

To approximate a set of examples the well known Ridge Regression procedure recommends finding the w which minimizes the function

$$a\|w\|^2 + \sum_{i=1}^{l}(y_i - w \cdot x_i)^2, \tag{5}$$

where a is a positive constant, called the *ridge parameter*. Notice that RR includes Least Squares as a special case (by setting $a = 0$). The RR prediction $\hat{y}_t$ for an example x_t is then $\hat{y}_t = w \cdot x_t$.

According to [22], the dual form RR formula for predicting the label y_t of a new input vector x_t is

$$Y(K + aI)^{-1}k, \tag{6}$$

where $Y = (y_1, \ldots, y_l)$ is the vector consisting of the labels of the examples in the training set, K is the $l \times l$ matrix of dot products of the input vectors $x_1, \ldots, x_l$ of those examples,

$$K_{j,i} = \mathcal{K}(x_j, x_i), \quad j = 1, \ldots, l, \quad i = 1, \ldots, l, \tag{7}$$

k is the vector of dot products of x_t and the input vectors of the training examples,

$$k_i := \mathcal{K}(x_i, x_t), \quad i = 1, \ldots, l, \tag{8}$$

and $\mathcal{K}(x, x')$ is the kernel function, which returns the dot product of the vectors x and x' in some feature space.

The non-conformity measure used by RRCP is

$$\alpha_i = |y_i - \hat{y}_i|, \tag{9}$$

where $\hat{y}_i$ is the prediction of the RR procedure for x_i based on the examples in (1). Using (6) the vector of all non-conformity scores $(\alpha_1, \ldots, \alpha_l)$ of the examples in (1) can be written in matrix form as

$$|Y - Y(K + aI)^{-1}K| = |Y(I - (K + aI)^{-1}K)|. \tag{10}$$

Furthermore $Y = (y_1, \ldots, y_{l-1}, 0) + (0, \ldots, 0, \tilde{y})$ and so the vector of non-conformity scores can be represented as $|A + B\tilde{y}|$ where

$$A = (y_1, \ldots, y_{l-1}, 0)(I - (K + aI)^{-1}K) \tag{11}$$

and

$$B = (0, \ldots, 0, \tilde{y})(I - (K + aI)^{-1}K). \tag{12}$$

Notice that now each $\alpha_i = \alpha_i(\tilde{y})$ varies piecewise-linearly as we change $\tilde{y}$. Therefore the p-value $p(\tilde{y})$ (defined by (2)) corresponding to $\tilde{y}$ can only change at the points where $\alpha_i(\tilde{y}) - \alpha_l(\tilde{y})$ changes sign for some $i = 1, \ldots, l-1$. This means that instead of having to calculate the p-value of every possible $\tilde{y}$, we can calculate the set of points $\tilde{y}$ on the real line that have a p-value $p(\tilde{y})$ greater than the given significance level δ, leading to a feasible prediction algorithm. A detailed description of this algorithm is given in [15], while a more efficient version can be found in [24].

4 Experiments and Results

We applied the Ridge Regression CP algorithm to three popular software effort estimation datasets the COCOMO 81 (COCOMO) the Desharnais and the ISBSG. The COCOMO [4] dataset contains information about 63 software projects from different applications. Each project is described by 17 cost attributes. The second dataset, Desharnais [5], includes observations for more than 80 systems developed by a Canadian Software Development House at the end of 1980. The third dataset, ISBSG [10] was obtained from the International Software Benchmarking Standards Group (ISBSG, Repository Data Release 9) and contains an analysis of software project costs for a group of projects. The projects come from a broad cross section of industry and range in size, effort, platform, language and development technique data. The release of the dataset used contains 100 characteristics and 3024 project data grouped in categories. Of those only 16 attributes (cost factors) were considered in our experiments, in which the dependent variable was the Full-Cycle Work Effort. In order to select these 16 attributes we performed several steps to clean, homogenize and codify the dataset. Firstly, we omitted columns describing attributes that may not be measured early in the development life cycle (i.e. are not available until the project concludes). Secondly, we removed those columns that had more than 40% of the total number of records filled with blank or unknown values. Thirdly, we kept only "qualitative" projects assessed as A, or B by the ISBSG reviewers and consistently reporting unique and easy to interpret information. Furthermore, all the numerical columns were cleaned from null values (row filtering), while we defined new categories to describe different but logically similar sample values of categorical columns, thus merging and homogenizing similar pieces of information into new categories (e.g. Oracle v7, Oracle 8.0 into Oracle). Finally, in order to be used with RR, every categorical (including multi-valued) attribute was replaced with n binary attributes, one for each category indicating whether the attribute belongs to that category or not. The final ISBSG dataset after this processing contained 467 records.

Before conducting our experiments the numerical attributes of all datasets were transformed to their natural logarithm values and then normalised to a minimum value of 0 and a maximum value of 1 so that all attributes lie in the same range and thus have the same impact. The effort values supplied to the algorithm were also log transformed, but the outputs produced were transformed back to the original effort scale before being compared with the actual values. Therefore all results reported here are on the original scale of efforts.

Our experiments followed a 10 fold cross-validation process. Each data set was split randomly into 10 parts of almost equal size and our tests were repeated 10 times, each time using one of the 10 parts as the testing set and the remaining 9 as the training set. This procedure was repeated 100 times for each dataset and the results reported here are over all runs. The kernel

used for computing the matrix K in (7) and the vector k in (8) was the RBF kernel, which is defined as

$$\mathcal{K}(x_j, x_i) = exp(-\frac{\|x_j - x_i\|^2}{2\gamma^2}). \quad (13)$$

The parameter γ of the RBF kernel as well as the ridge parameter a in (6) are typically determined by trial and error. Thus for γ we tried the values 2 to 7 with increments of 0.5 and for a the values $\{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$.

We first applied the original RR approach to each dataset in order to check its performance on the data in question. The performance of the method was evaluated using the Mean Magnitude of Relative Error ($MMRE$), which is one of the most widely used metrics for evaluating the accuracy of cost estimation models. The Magnitude of Relative Error for a test project i was calculated as

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \quad (14)$$

where y_i is the actual effort for project i and $\hat{y}_i$ is the predicted effort produced by (6) with x_i as the new input vector. The MRE of all projects in each of the 10 parts of every dataset was calculated using only the projects in the remaining 9 parts as training examples. This process resulted in one MRE value for each project, and since it was repeated 100 times, the $MMRE$ for each dataset was calculated as the mean of all $100N$ MREs where N is the number of projects in that dataset. Table 1 reports the best results obtained together with the γ value and the ridge parameter a that were used. The results of this table suggest that a considerable accuracy in predicting software effort values has been achieved by the base method, which is comparable to what the international literature has to offer using similar techniques up until today.

After the encouraging results obtained with the original Ridge Regression method we applied the RRCP to the three datasets. More specifically, RRCP was applied for the 95%, 90% and 80% confidence levels to every project in each of the 10 parts of the dataset, using as training set the projects in the remaining 9 parts. This process resulted in three predictive intervals for every project, one for each of the three confidence levels. Again the same process was repreated 100 times and thus resulted in $100N$ predictive intervals for

Dataset	γ	a	$MMRE$
COCOMO	5.5	0.001	0.4221
Desharnais	5	0.05	0.3454
ISBSG	3.5	0.1	0.5969

Table 1 The best results of the original Ridge Regression method and the parameters used.

Data Set	Median Width			Median Relative Width			Percentage of Errors (%)		
	80%	90%	95%	80%	90%	95%	80%	90%	95%
COCOMO	170.8	278.8	383.4	1.3593	2.2075	3.1105	19.11	8.68	3.68
Desharnais	4549	5579	6396	1.2588	1.5462	1.7698	19.03	9.06	4.38
ISBSG	4451	6571	9126	1.6821	2.4639	3.3747	20.01	10.02	5.01

Table 2 The tightness and reliability results of the Ridge Regression CP on the three data sets.

each confidence measure, where N is the number of projects in the dataset in question. The parameters used for these experiments are the same as the ones used for the original RR method, given in table 1.

Table 2 reports the results of the RRCP in terms of the tightness and reliability of the produced predictive intervals. The first part of the table reports the median widths of the intervals for each of the three confidence levels (95%, 90% and 80%). In addition to the width of each predictive interval, we also calculated its Relative Width (RW) as $RW_i = w_i/y_i$ where w_i is the width of the predictive interval produced for the example x_i and y_i is its actual effort value. The median values of the relative widths can be found in the second part of the table. We chose to report the median values of both the widths and the relative widths instead of the means so as to avoid the strong impact of a few extremely large or extremely small intervals. In the third and final part of table 2 we check the reliability of the obtained predictive intervals. This is performed by reporting the percentage of examples for which the true effort value is not inside the interval output by the RRCP. In effect this checks empirically the validity of the predictive intervals produced. The percentages reported here are either below or almost equal to the required significance level.

The enhancement of the original method with CP offers a much more informative scenery for a project manager to decide how to distribute effort as he/she is provided with lower and upper estimation limits instead of single effort prediction values, something that enables him/her to plan according to worst and best case scenarios. It is worth to note that the median widths of the predictive intervals reported in Table 2 for the 95% confidence level correspond to 3.36%, 27.34% and 6.09% of the whole range of efforts of the COCOMO, Desharnais and ISBSG datasets respectively. This provides a strong indication that the intervals produced are narrow enough to be useful in practice.

5 Conclusions

We have proposed the use of the Ridge Regression Conformal Predictor for obtaining reliable software effort confidence intervals. Confidence intervals

make an estimate much more informative than point predictions since they indicate the level of uncertainty of the estimate; the bigger the interval for a given project the higher the uncertainty of the estimate. This allows for different treatment of estimates, so that the uncertain estimates are given further thought or more careful planning. The main advantages of CPs over other techniques that produce confidence intervals are that (a) the confidence intervals produced by CPs are provably valid, (b) they do not require any extra assumptions other than i.i.d. and (c) they produce a different confidence interval for each individual project, the size of which reflects how difficult the effort of the project is to estimate.

The RRCP was applied on three empirical cost datasets each including a set of cost factors measured over a series of completed software projects. The results obtained by the proposed approach demonstrate that it can produce predictive intervals that are well-calibrated and narrow enough to be useful to project managers. Our plans for future work include utilising Conformal Prediction to produce confidence intervals for specific project cost descriptive variables, such as size, complexity and duration and by using historical data samples to attempt to associate these variables with effort. Furthermore, the approach followed in this paper may also be applied in conjunction with other algorithms reported in literature to perform well in predicting effort, such as neural networks and genetic algorithms. In this case the Ridge Regression part will be substituted by other predictive models to form the basis for Conformal Prediction. Finally, genetic algorithms can be utilized for the selection of the most appropriate subset of cost factors to be used as inputs to the CP so as to improve even further the results reported here.

References

1. Andreou, A., Papatheocharous, E.: Tools in Artificial Intelligence, chap. 1. Computational Intelligence in Software Cost Estimation: Evolving conditional sets of effort value ranges, pp. 1–20. I-Tech Education and Publication KG, Vienna, Austria (2008). URL http://intechweb.org/downloadpdf.php?id=5277
2. Angelis, L., Stamelos, I.: A simulation tool for efficient analogy based cost estimation. Empirical software engineering **5**, 35–68 (2000)
3. Bellotti, T., Luo, Z., Gammerman, A., Delft, F.W.V., Saha, V.: Qualified predictions for microarray and proteomics pattern diagnostics with confidence machines. International Journal of Neural Systems **15**(4), 247–258 (2005)
4. Boehm, B.: Software Engineering Economics. Prentice Hall (1981)
5. Desharnais, J.M.: Analyse statistique de la productivite des projects de development en informatique a partir de la technique de points de fonction. MSc. Thesis. Montreal Universite du Quebec (1988)
6. Gammerman, A., Vapnik, V., Vovk, V.: Learning by transduction. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 148–156. Morgan Kaufmann, San Francisco, CA (1998)
7. Gammerman, A., Vovk, V., Burford, B., Nouretdinov, I., Luo, Z., Chervonenkis, A., Waterfield, M., Cramer, R., Tempst, P., Villanueva, J., Kabir, M.,

Camuzeaux, S., Timms, J., Menon, U., Jacobs, I.: Serum proteomic abnormality predating screen detection of ovarian cancer. The Computer Journal, doi:10.1093/comjnl/bxn021 (2008)
8. Gruschke, T., Jørgensen, M.: The role of outcome feedback in improving the uncertainty assessment of software development effort estimates. ACM Transactions of Software Engineering Methodology **17**, 1–35 (2008)
9. Huang, X., Ho, D., Ren, J., Capretz, L.: Improving the COCOMO model using a neuro-fuzzy approach. Applied Soft Computing **7**, 9–40 (2007)
10. International Software Benchmarking Standards Group: The ISBSG estimating, benchmarking & research suite release 9 (2005). URL http://www.isbsg.org/
11. Jørgensen, M., Shepperd, M.: A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering **33**(1), 33–53 (2007)
12. Jørgensen, M., Sjøberg, D.: An effort prediction interval approach based on the empirical distribution of previous estimation accuracy. Information Software and Technology **45**, 123–136 (2003)
13. Jørgensen, M., Teigen, K., Moløkke, K.: Better sure than safe? overconfidence in judgment based software development effort prediction intervals. Systems and Software **70**, 79–93 (2004)
14. Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., Webster, S.: An investigation of machine learning based prediction systems. Journal of Systems and Software **53**(1), 23–49 (2000)
15. Nouretdinov, I., Melluish, T., Vovk, V.: Ridge regression confidence machine. In: Proceedings of the 18th International Conference on Machine Learning (ICML'01), pp. 385–392. Morgan Kaufmann, San Francisco, CA (2001)
16. Nouretdinov, I., Vovk, V., Vyugin, M.V., Gammerman, A.: Pattern recognition and density estimation under the general i.i.d. assumption. In: Proceedings of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory, *Lecture Notes in Computer Science*, vol. 2111, pp. 337–353. Springer (2001)
17. Oliveira, A.: Estimation of software projecs effort with support vector regression. Neurocomputing **69**(13-15), 1749–1753 (2006)
18. Papadopoulos, H.: Tools in Artificial Intelligence, chap. 18. Inductive Conformal Prediction: Theory and Application to Neural Networks, pp. 315–330. I-Tech, Vienna, Austria (2008). URL http://intechweb.org/downloadpdf.php?id=5294
19. Papadopoulos, H., Gammerman, A., Vovk, V.: Normalized nonconformity measures for regression conformal prediction. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2008), pp. 64–69. ACTA Press (2008)
20. Papatheocharous, E., Andreou, A.: Software cost estimation using artificial neural networks with inputs selection. In: Proceedings of the 9th International Conference on Enterprise Information Systems, pp. 398–407. Madeira, Funchal (2007)
21. Proedrou, K., Nouretdinov, I., Vovk, V., Gammerman, A.: Transductive confidence machines for pattern recognition. In: Proceedings of the 13th European Conference on Machine Learning (ECML'02), *Lecture Notes in Computer Science*, vol. 2430, pp. 381–390. Springer (2002)
22. Saunders, C., Gammerman, A., Vovk, V.: Ridge regression learning algorithm in dual variables. In: Proceedings of the 15th International Conference on Machine Learning (ICML'98), pp. 515–521. Morgan Kaufmann, San Francisco, CA (1998)
23. Saunders, C., Gammerman, A., Vovk, V.: Transduction with confidence and credibility. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, vol. 2, pp. 722–726. Morgan Kaufmann, Los Altos, CA (1999)
24. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer, New York (2005)