

Column Pruning Beats Stratification in Effort Estimation

Omid Jalali, Tim Menzies, Dan Baker
Lane Dept. CS & EE
West Virginia University, USA

ojalali@mix.wvu.edu, tim@menzies.us, dbaker6@mix.wvu.edu

Jairus Hihn ^{*†}
Jet Propulsion Laboratory,
Pasadena, California, USA

jairus.m.hihn@jpl.nasa.gov

Abstract

Local calibration combined with stratification, also known as row pruning, is a common technique used by cost estimation professionals to improve model performance. The results presented in this paper raise several serious questions concerning the benefits of row pruning for improving effort estimation indicating the need to rethink standard practice. Firstly, the mean size of improvements from row pruning appears to be relatively small compared to the size of the standard deviations in effort estimation data. Secondly, the advantages of row pruning especially for the purposes of deleting spurious outliers can be achieved using column pruning much more effectively. Hence, we advise against row pruning and advocate column pruning instead.

1. Introduction

Many believe that software effort models are a waste of time. Jorgensen reports that the majority of software effort estimates are expert-based and are not automatically generated from models [12]. Effort estimation requires generalizing from a small number of old projects. Generalization from such limited experience is an inherently under-constrained problem. Hence, the learned effort models can exhibit large deviations. For example, we reported at PROMISE 2006 that the standard deviation on the mean magnitude of the relative error, or MRE¹, of effort estimations in 28 data sets was enormous:

$$\{\min\%, \text{median}\%, \max\%\} = \{45\%, 157\%, 649\%\}$$

^{*}The research described in this paper was carried out at West Virginia University and the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, funded by the NASA Software Assurance Research Program. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

[†]See <http://menzies.us/pdf/07micrococomo.pdf> for an earlier draft of this paper.

¹MRE = $\text{abs}(\text{predicted} - \text{actual})/\text{actual}$.

Such large deviations confuse both the generated estimates and attempts to compare one effort estimation method with another. For example, Chulani & Boehm [7] and Shepperd & Schofield [24] report that *row pruning*, also known as *stratification*, of training data into related subsets improved estimation PRED² by 5% to 12% and 4% to 44% (respectively). When compared to standard deviations in the range 45% to 649%, the estimation deltas reported by Chulani & Boehm and Shepperd & Schofield may be statistically insignificant.

Regardless of their drawbacks, various institutions demand that managers produce model-based estimates:

- When lobbying for resources, or arguing against funding cuts, model-based methods are useful tools to develop, certify, and defend an estimate.
- Models not only let us make decisions; they let us audit them as well. Numerous American government bodies (DoD, NASA) require audit-able effort estimates.
- Models let us “look before we leap”; e.g. effort models allow a manager to explore and assess numerous alternate scenarios in the planning stages of a project.
- Models can help train newcomers who can browse models to learn the processes of an organization.

Since model-based estimates are so useful, and since the deviation problem undermines the authority of those estimates, we have been exploring methods for reducing the deviation problem. Miller shows theoretically that decreasing the number of variables decreases the deviation of a linear model learned from those variables by minimizing least squares error [20]. That is, the fewer the variables, the more restrained are the model predictions. Inspired by Miller, and certain empirical results [10, 14], we developed the COSEKMO workbench [18]. This COCOMO-based tool [1, 2] combines variable pruning with numerous alternate model generation models (local calibration, linear regression, model trees). COSEKMO assessed these alternatives via their mean performance as well as several other modeling criteria (reduced deviation, increased correlation,

²PRED(N) is the percent of test cases with an MRE less than N%.

increased PRED(30)). In the results consistent with Miller’s theoretical prediction, the standard deviation on the MRE% in 28 data sets fell dramatically:

$$\{min, median, max\} : \overbrace{\{45, 157, 649\}}^{LC} \implies \overbrace{\{20, 39, 100\}}^{COSEEKMO}$$

(LC is the *local calibration* method that is standard practice in the COCOMO community.)

Since the deviation was reduced, it was possible to compare different methods and select better alternatives. Hence, the mean MRE% also decreased:

$$\{min, median, max\} : \overbrace{\{43, 58, 188\}}^{LC} \implies \overbrace{\{22, 38, 64\}}^{COSEEKMO}$$

In summary, COSEEKMO implements an elaborate comparison between numerous methods. Thirty times, the order of data in a training set is randomized, ten test cases are removed, and numerous techniques are applied to the remaining cases to build an effort estimator. This estimator is then applied to the ten test cases and comparative statistics are collected for each method. Model selection heuristics are then applied to return the preferred model (see Figure 1). A completing justification of the design decisions made by COSEEKMO is beyond the scope of this paper. For full details, see [18].

COSEEKMO is slow to run and after much experimentation, we have learned that most of that runtime is wasted. For example, COSEEKMO’s model trees [23] use 35% of the CPU time yet rarely return the preferred model. Also, while certain methods are clearly inferior (e.g. LC), most of the remaining methods all have very similar performance.

Therefore, we are currently designing μ COCOMO (micro-COCOMO) - the subset of methods within full COSEEKMO that are the simplest to build, perform better than LC, perform as well as full COSEEKMO, dramatically reduce the large deviations, scale to larger problems, and which can be used early in the life cycle of an effort model.

An interesting result of COSEEKMO is that it strongly endorses Boehm’s 1981 research into COCOMO. Over the last three years, (starting with [19]), we have tried to improve COCOMO using modern data mining tools. To date, we have found no extension or variation that performs significantly better.

There is only one area where we offer a clarification to Boehm’s work. Prior results about the benefits of row pruning can no longer be uncritically accepted. For example, Chulani & Boehm [7], report that row pruning (stratification) plus local calibration improved their mean PRED values by 5% to 12%. Given the large deviations seen in the top sd(MRE) plot of Figure 6, such small improvements may be due to simple noise. We will show that based on the NASA93 data set, the advantages of row pruning (deleting

of spurious outliers) can be achieved using column pruning. Further, in the case where only some of the variables are noisy, column pruning can delete part of a row while preserving the rest of the information in a row.

This report discusses experiments that assess the merits of adding pruning operators into μ COCOMO. The original version of COSEEKMO just pruned columns (variables). An alternate strategy is to augment or replace column pruning with row pruning. The motivation for row pruning, also known as *stratification*, is very clear. In a COCOMO data set, each row is one project and data collected in one context (e.g. NASA flight software) may not be completely relevant to some other context. In theory, similar projects have less variation and so can be easier to calibrate. Nevertheless, we show here that in many situations, row pruning offers no advantage over other methods. Hence, μ COCOMO will perform column pruning, but not row pruning.

2. Background

COSEEKMO is based around COCOMO. The core intuition behind COCOMO is as a program grows in size, the development effort grows exponentially. More specifically:

$$effort(personmonths) = a * (KLOC^b) * \left(\prod_j EM_j \right)$$

Here, $KLOC$ is thousands of delivered source instructions, EM_j is one of Figure 2 effort multipliers with precise values from Figure 3, and a and b are the parameters tuned by Boehm’s local calibration (LC) procedure of Figure 4.

We use COCOMO since, unlike other models such as PRICE-S [21], SLIM [22], or SEER-SEM [11], it is an open model with published data. All details were published in Boehm’s text *Software Engineering Economics* [1,2]. There are two version of COCOMO: COCOMO-I and COCOMO-II. We base our studies on COCOMO-I data since we know of no large public domain COCOMO-II data sets. In contrast, several COCOMO-I data sets are available in the PROMISE repository [4]:

- *COC81*: The 63 projects used to define COCOMO-I. This data comes from a variety of domains including engineering, science, financial, etc.
- *NASA93*: The 93 NASA projects from the 1980s and 1990s. Since this data comes from NASA, it is stratified to just aerospace applications.

3. External Validity

The external validity of our study must be discussed. These results are only relevant to COCOMO-style paramet-

```

FUNCTION worse(x,y)
IF statisticallyDifferent(x,y) THEN
  IF error(x) < error(y) THEN RETURN y FI rule1
  IF error(y) < error(x) THEN RETURN x FI rule1
ELSE
  IF sd(x) < sd(y) THEN RETURN y FI rule2
  IF sd(y) < sd(x) THEN RETURN x FI rule2

  IF correlation(x) < correlation(y) THEN RETURN x FI rule3
  IF correlation(y) < correlation(x) THEN RETURN y FI rule3

  IF pred(x) < pred(y) THEN RETURN x FI rule4
  IF pred(y) < pred(x) THEN RETURN y FI rule4

  IF |Subset(x)| < |Subset(y)| THEN RETURN y FI rule5
  IF |Subset(y)| < |Subset(x)| THEN RETURN x FI rule5
FI
RETURN 0 if no reason to return true

```

Figure 1. COSEEKMO rejection rules (slightly updated from [18]). *Error* refers to MMRE. *Worse's* statistically difference test compares two MMREs *x* and *y* using a two-tailed t-test at the 95% confidence interval.

upper: increase these to decrease effort	ACAP: analysts capability PCAP: programmers capability AEXP: application experience MODP: modern programming practices TOOL: use of software tools VEXP: virtual machine experience LEXP: language experience
middle	SCED: schedule constraint
lower: decrease these to increase effort	DATA: data base size TURN: turnaround time VIRT: machine volatility STOR: main memory constraint TIME: time constraint for cpu RELY: required software reliability CPLX: process complexity

Figure 2. COCOMO 81 effort multipliers.

		very low	low	nominal	high	very high	extra high
upper (increase these to decrease effort)	ACAP	1.46	1.19	1.00	0.86	0.71	
	PCAP	1.42	1.17	1.00	0.86	0.70	
	AEXP	1.29	1.13	1.00	0.91	0.82	
	MODP	1.2	1.10	1.00	0.91	0.82	
	TOOL	1.24	1.10	1.00	0.91	0.83	
	VEXP	1.21	1.10	1.00	0.90		
	LEXP	1.14	1.07	1.00	0.95		
middle	SCED	1.23	1.08	1.00	1.04	1.10	
lower (increase these to increase effort)	DATA		0.94	1.00	1.08	1.16	
	TURN		0.87	1.00	1.07	1.15	
	VIRT		0.87	1.00	1.15	1.30	
	STOR			1.00	1.06	1.21	1.56
	TIME			1.00	1.11	1.30	1.66
	RELY	0.75	0.88	1.00	1.15	1.40	
	CPLX	0.70	0.85	1.00	1.15	1.30	1.65

Figure 3. Precise values for Figure 2.

ric effort estimation. They are based only on the projects de-

The matrix $D_{i,j}$ holds

- the natural logs of the *KLOC* estimates,
- the natural logs of the actual efforts for projects $i \leq j \leq t$,
- the natural logs of the cost drivers (the scale factors and effort multipliers) at locations $1 \leq i \leq 15$ (for COCOMO 81) or $1 \leq i \leq 22$ (for COCOMO-II).

The following calculation yields estimates for “*a*” and “*b*” that minimizes the sum of the squares of residual errors between predicted and actual efforts in a COCOMO model.

$$\begin{aligned}
 EAF_i &= \sum_j^N D_{i,j} \\
 a_0 &= t \\
 a_1 &= \sum_i^t KLOC_i \\
 a_2 &= \sum_i^t (KLOC_i)^2 \\
 d_0 &= \sum_i^t (actual_i - EAF_i) \\
 d_1 &= \sum_i^t ((actual_i - EAF_i) * KLOC_i) \\
 b &= (a_0 d_1 - a_1 * d_0) / (a_0 a_2 - a_1^2) \\
 a_3 &= (a_2 d_0 - a_1 d_1) / (a_0 a_2 - a_1^2) \\
 a &= e^{a_3}
 \end{aligned}$$

Figure 4. LC: local calibration (from [1]).

scribed in the *COC81* and *NASA93* data sets. Nevertheless, to the best of our knowledge, *COC81* and *NASA93* represent the universe of publicly available COCOMO data sets.

Also, our results are limited to the space of learners, WRAPPERS, and other methods explored by COSEEKMO and μ COCOMO, which are:

- local calibration,
- model trees,
- linear regression,
- LocalWRAPPER, and
- Euclidean nearest neighbor.

It is possible that other methods would outperform and overturn our results. In the true spirit of PROMISE, all our data and tools are on-line. We invite other researchers to download them, extend them, and to supersede our results.

4. Experiments

4.1 Prior Row Pruning Experiments

Our previous experiments were quite negative on the merits of row pruning. As reported in [18], the rows of *NASA93* divide into various subsets:

- All the records,
- Whether or not it is flight or ground system,
- The parent project that includes the project expressed in this record,
- The NASA center where the work was conducted,
- etc.

Some of these subsets are bigger than others. Given a record labeled with subsets $\{S_1, S_2, S_3, \dots\}$ then we say a *candidate stratification* has several properties:

- It contains records labeled $\{S_i, S_j\}$.
- The number of records in S_j is greater than S_i ; i.e. S_j is the superset and S_i is the stratification.
- S_j contains 150% (or more) of records in S_i .
- S_i contains at least 20 records.

COSEEKMO searched for effort models that were better in the subset than the superset. *NASA93* contains 207 candidate stratifications (including one stratification containing all the records), and better effort models were found in only four cases. That is, in the overwhelming majority of cases, row pruning (stratification) offered no advantage.

4.2 New Row Pruning Experiments

Another possibility, not previously explored, is that the stratifications are mis-labeled. The stratifications seen in *COC81* and *NASA93* come from row labels. If domain experts do not fully understand the factors that affect software costs, they might inadvertently group together the wrong rows.

To test that possibility, we performed row pruning experiments where stratifications were selected from the training set via their Euclidean distance to the test case. That is, for each member of the test set, an effort model is generated (using LC) from the closest N members of the training set. For this experiment, *distance* was defined using:

$$\text{distance case1 to case2} = \sqrt{\sum_{X \in EM} \left(\frac{X_1 - X_2}{\max(X) - \min(X)} \right)^2}$$

(where the X values are the numeric values of Figure 3).

Figure 5 shows the effects of different combinations of row and column pruning on the standard deviation of the MRE, the mean MRE, and PRED(30). This is our standard report format so, before reviewing those findings, we review the details of those reports:

- The x-axis of Figure 5 denotes which subset of *COC81* or *NASA93* was used in this experiment.
- Plots are always shown in sets of three. A vertical line through the three plots would show results from the same subset.
- The plots were generated using the COSEEKMO method described in the introduction.
- For $sd(MRE)$ and $mean(MRE)$, the goal is to get *lower* lines than the other methods.
- For $\%pred(30)$, the goal is to get *higher* lines than the other methods.

As to the particulars of Figure 5:

- These plots show results from some preprocessing of the data followed by local calibration (LC). In the case of Figure 5, those preprocessors were some combination of row and column pruning.
- In this plot, LC used the precise effort multiplier values of Figure 3.
- The lines labeled “Precise-None-None-LC” show the standard effort estimation method as recommended by Boehm; i.e. just perform local calibration.
- The top two lines of each plot show a comparison between row pruning and no row pruning when there is no column pruning present.
- The last two lines add in column pruning. For this figure, the column pruning method was based on Kohavi’s WRAPPER [15]. The WRAPPER passes different subsets of variables to some oracle (in our case, LC) and returns the subset which yields the best performance. WRAPPER is thorough but can be slow since (in the worst case), it has to explore all subsets of the available variables.
- The row pruning in these results used the $N = 20$ closest cases to build an effort model for the test cases (and the impact of changing N is discussed below).

In Figure 5, the following effect is a strong argument that local calibration, by itself, needs to be augmented or replaced:

- The “Precise-None-None-LC” results in the top $sd(MRE)$ plot show devastating large deviations resulting from standard methods (LC) - often over 100% and sometimes much larger.

The following effects are strong arguments against row pruning via Euclidean distance measures:

- The top two plots for $sd(MRE)$ and $mean(MRE)$ show that without other pruning methods, row pruning does no better than LC.
- The top two plots also show that when column pruning is active, row pruning adds little or no advantage.

The following effects are strong arguments for column pruning:

- The top two plots show that the column pruning always resulted in much less error and deviation.
- In the middle plot, column pruning without row pruning generates the smallest errors.

The only argument that can be mounted from these results against column pruning is:

- In the bottom plot, column pruning, by itself, sometimes yields the lowest PRED. This effect is much less marked than those discussed above. So, it does not dissuade against our thesis that there is much merit in column pruning and little merit in row pruning.

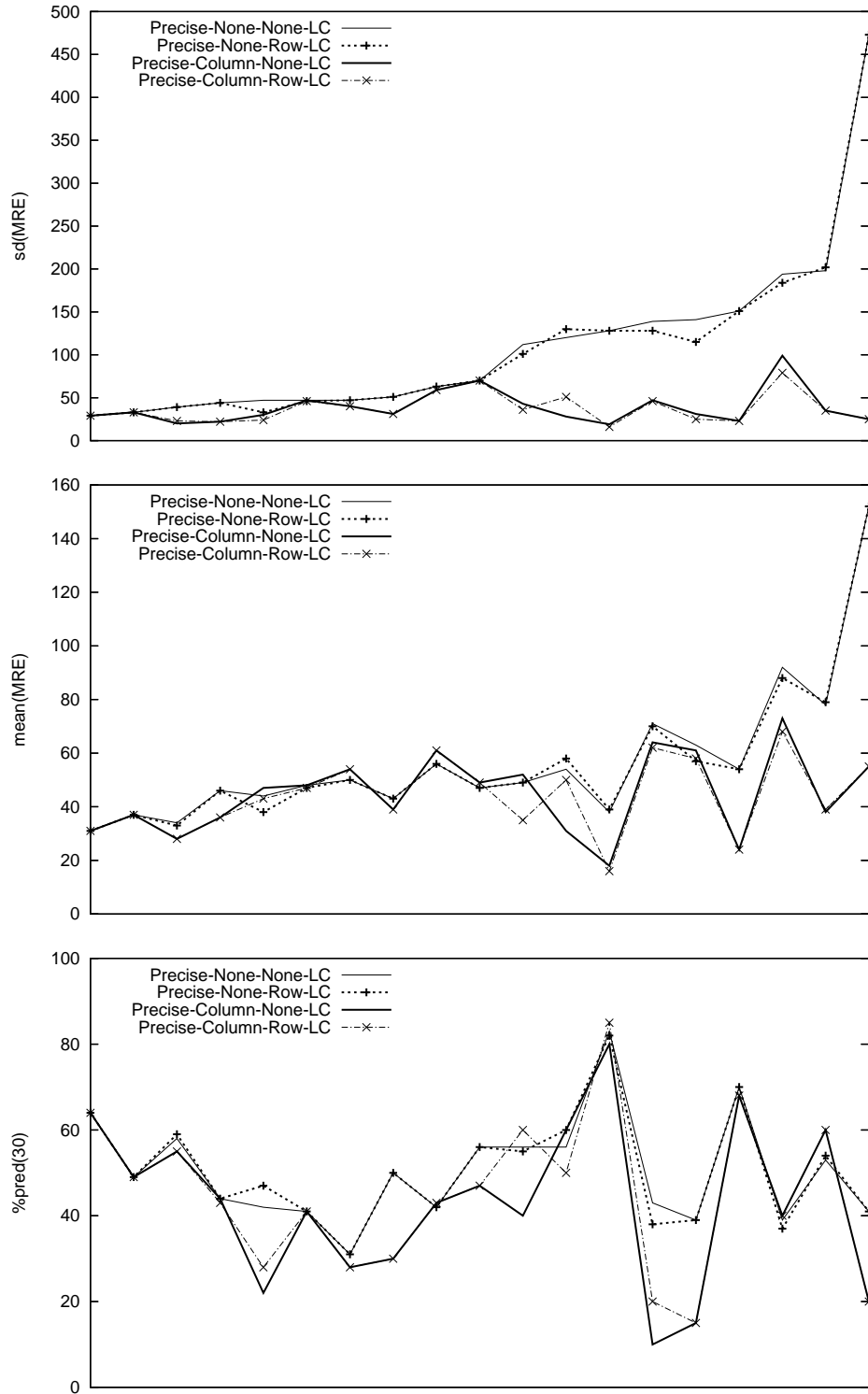


Figure 5. Effects of different pruning tactics.

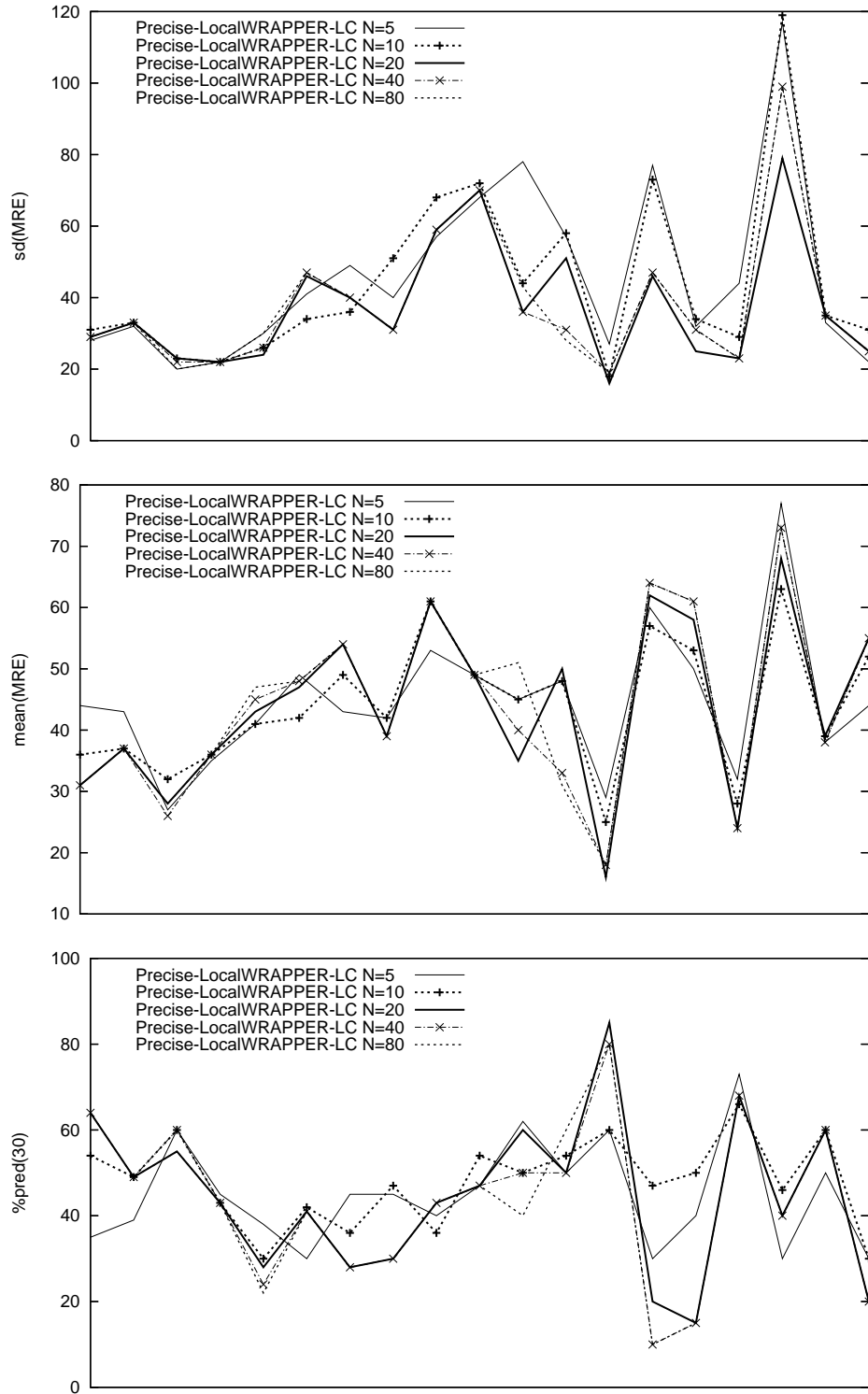


Figure 6. Effects of different neighborhoods.

These results run counter to several other publications (e.g. [7, 24]) including our research [6, 19] that report an improvement in PRED following stratification. One possibility is that the above results did not select the right number of neighbors to build an effort estimation model.

Figure 6 tests that speculation. The proceeding discussion endorsed column pruning with the WRAPPER followed by LC. Figure 6 augments such a rig with a nearest-neighbor section of $N \in \{5, 10, 20, 40, 80\}$. No clear “best” nearest-neighbor is apparent. Indeed, looking at the mean(MRE) results, changing the size of neighborhood had very little effect on the mean error. All in all, there is no evidence in Figure 6 that the above results are conflated by a wrong selection of nearest neighbors.

5. Discussion

Based on the above, we assert that:

1. Column pruning is particularly useful at reducing the mean and standard deviation of the MRE seen with COCOMO-style local calibration. This effect has been reported previously [18].
2. Row pruning offers no advantage for the generation of effort models using a COCOMO-style LC procedure. This is a new result.

Conclusion #2 is counter-intuitive. Many researchers (e.g. [6, 7, 19, 24]) advise that a standard practice for modeling is to “remove outliers”; i.e. cull rows of data that contain spurious signals. We speculate that prior results regarding the merits of stratification may have been conflated by the large deviation effect; i.e. their reported improvements were merely noise within a widely variant space of solutions.

Another possibility is that the “remove outliers” advice is subtly misguided. In the case where the outliers are caused by a small number of deviant variables, then if those variables are removed (by column pruning), the information in the rest of the row remains and can assist in the modeling task. In that case, the “remove outlier” advantage seen in row pruning would also be achieved with column pruning.

Our own prior results that endorsed row pruning need further discussion. Chen, Menzies, Port, and Boehm (hereafter, CMPB) [6] concluded that “pruning always improved estimation effectiveness”. That conclusion was made based on the results in Figure 7. The y-axis of that figure shows the PRED(30) results seen after performing 30 linear regressions on $\frac{2}{3}$ -rds of the data (random selected), then applying the learned model to the remaining $\frac{1}{3}$ -rd of the data. The top *before* plot shows the linear regression results while the *after* plot shows the effect of linear regression plus column pruning via the WRAPPER. The data sets on the x-axis are sorted left-to-right according to how many records they contain (see the blue plot). To produce those

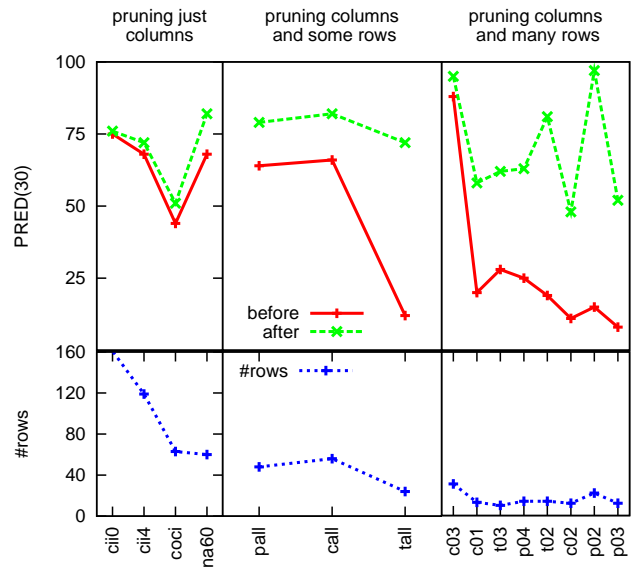


Figure 7. Results from [6].

data sets, the four data sets shown on the left hand side (*cii0*, *cii4*, *coci*, *na60*) were divided two ways:

1. The data was divided into the project, geographical center, and task performed by the software. This generated the three middle data sets (*pall*, *call*, *tall*).
2. The data was divided according to their most specific stratification; e.g. a project from a particular project at a particular center working on a particular task. This generated the eight smallest data sets shown on the right (*c03*, *c01*, *t03*, *p04*, *t02*, *c02*, *p02*, *p03*).

That is, the data sets in the middle and on the right of Figure 7 are stratifications (row prunings) of data shown on the left.

From those results, CMPB concluded a column pruning, plus extensive row pruning, was the best effort estimation policy. However, those results applied row and column pruning to non-identical portions of data sets when comparing different combinations of column and row pruning.

In contrast, the analysis discussed here compares row pruning separately and in combination with column pruning. It also uses identical data sets in all these comparisons. Hence, this paper is a more complete analysis of row pruning and column pruning effects.

6. Conclusion

From our results, we can make several comments about certain prior results [6, 18].

6.1 Chen, Menzies, Port, & Boehm '05 [6]

The estimation improvements attributed to row pruning by Chen et.al. [6] were not found here. Those results were due to a methodological error: i.e. row and column pruning was not applied consistently to that data. Here, after applying column and row pruning to the same data, we found that row pruning gave no improvement over column pruning.

6.2 Menzies, Chen, Hihn, & Lum '06 [18]

What do these results say about the COSEEKMO study [18]? Is it a needlessly elaborate system?

We would argue for the utility of COSEEKMO. COSEEKMO is a *certification* environment and μ COCOMO is a *production* environment. COSEEKMO is a general rig where new ideas and tools can be rigorously and exhaustively tested. However, it is also the flame that burns away the fat. Using COSEEKMO, we can find lean and fast methods for effort estimation, which can simplify the work of analysts performing model-based estimation. Our current plan is to code all those lean and fast methods into μ COCOMO.

The design process of μ COCOMO imposed an audit of effort modeling methods. Such an audit is long overdue. Many effort estimation methods such as clustering [9], neural networks [17], and case-based reasoning [24] exist in the literature. In industry, the most commonly-used formal techniques are parametric or regression-based techniques, as used in COCOMO-I [1], COCOMO-II [2], SLIM [22], SEER-SEM [11], and PRICE-S [21]. There are very few published studies that empirically compare this diverse set. What is more usual are narrowly focused studies, such as those conducted by Kemerer [13], Briand [5], Lum [16], or Ferens [8] that test linear regression models in different environments.

The reason for this lack of empirical studies is two-fold. As mentioned in the introduction, most effort estimation is expert-based, not model-based, and expert-based evaluations are difficult to repeat. Also, the large deviation problem makes it very difficult to come to definitive conclusions about the relative merits of different model-based methods.

COSEEKMO's ability to reduce that variance should change the nature of empirical effort estimation research. It is now possible to compare, and reject, certain model-based methods as (i) inferior or (ii) competitive in terms of MRE or PRED, but inferior in terms of implementation complexity or runtime.

We therefore anticipate a few years of COSEEKMO/ μ COCOMO-style experimentation where model-based methods are debated before some consensus emerges on best-practices for model-based effort estimation. Once that consensus is available, the field can

move on to comparing our best model-based methods with expert-based estimation.

7. Future Work

It is possible that stratification to particular project types would improve effort estimation. What we have shown in our work so far is that, to date, we have not found that the current COCOMO attributes can characterize those "particular project types". It is possible that further details from the target domain could be used to improve effort estimation; e.g. when Boetticher augmented the standard effort estimation attributes with knowledge of user-interface complexity, the resulting models had impressively large PRED values [3].

Therefore, over summer 2007, we plan to experiment with augmented COCOMO data sets that include more domain-specific descriptors of projects. The hypothesis is that the information in these additional columns will improve effort estimation; i.e. the row pruning based on these extra columns will decrease MRE and increase PRED.

In addition to the above, the authors of this paper are exploring other measures, such as non-parametric methods to evaluate the experiment results, to check and verify the stability of these conclusions.

References

- [1] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.
- [2] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.
- [3] G. Boetticher. When will it be done? the 300 billion dollar question, machine learner answers. *IEEE Intelligent Systems*, June 2003.
- [4] G. Boetticher, T. Menzies, and T. Ostrand. The PROMISE Repository of Empirical Software Engineering Data, 2007. <http://promisedata.org/repository>.
- [5] L. Briand, T. Langley, and I. Wiczorek. A replicated assessment and comparison of common software cost modeling techniques. In *Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland*, pages 377–386, 2000.
- [6] Z. Chen, T. Menzies, D. Port, and B. Boehm. Finding the right data for software cost modeling. *IEEE Software*, Nov 2005.
- [7] S. Chulani, B. Boehm, and B. Steece. Bayesian analysis of empirical software engineering cost models. *IEEE Transaction on Software Engineering*, 25(4), July/August 1999.
- [8] D. Ferens and D. Christensen. Calibrating software cost models to Department of Defense Database: A review of ten studies. *Journal of Parametrics*, 18(1):55–74, November 1998.

- [9] M. Garre, M. S. J.J. Cuadrado-Gallego, M. Charro, and D. Rodriguez. Segmented parametric software estimation models: Using the em algorithm with the isbsg 8 database. In *27th International Conference on Information Technology Interfaces. ITI 2005, Dubrovnik, Croatia, 2005*.
- [10] M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions On Knowledge And Data Engineering*, 15(6):1437– 1447, 2003.
- [11] R. Jensen. An improved macrolevel software development resource estimation model. In *5th ISPA Conference*, pages 88–92, April 1983.
- [12] M. Jorgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2):37–60, 2004.
- [13] C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, May 1987.
- [14] C. Kirsopp and M. Shepperd. Case and feature subset selection in case-based software project effort prediction. In *Proc. of 22nd SGAI International Conference on Knowledge-Based Systems and Applied Artificial Intelligence, Cambridge, UK, 2002*.
- [15] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
- [16] K. Lum, J. Powell, and J. Hihn. Validation of spacecraft cost estimation models for flight and ground systems. In *ISPA Conference Proceedings, Software Modeling Track*, May 2002.
- [17] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. S. ofield1, M. Shepperd, and S. Webster. An investigation of machine learning based prediction systems. *The Journal of Systems and Software*, 53(1):23–29, 2000.
- [18] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, November 2006. Available from <http://menzies.us/pdf/06coseekmo.pdf>.
- [19] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes. Validation methods for calibrating software effort models. In *Proceedings, ICSE, 2005*. Available from <http://menzies.us/pdf/04coconut.pdf>.
- [20] A. Miller. *Subset Selection in Regression (second edition)*. Chapman & Hall, 2002.
- [21] R. Park. The central equations of the price software cost model. In *4th COCOMO Users Group Meeting*, November 1988.
- [22] L. Putnam and W. Myers. *Measures for Excellence*. Yourdon Press Computing Series, 1992.
- [23] J. R. Quinlan. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992. Available from <http://citeseer.nj.nec.com/quinlan92learning.html>.
- [24] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12), November 1997. Available from http://www.utdallas.edu/~rbaner/SE_XII.pdf.