The Evaluation of Weighted Moving Windows for Software Effort Estimation

Sousuke Amasaki[1] and Chris Lokan[2]

[1] Okayama Prefectural University,
Department of Systems Engineering
amasaki@cse.oka-pu.ac.jp
[2] UNSW Canberra,
School of Engineering and Information Technology
c.lokan@adfa.edu.au

Abstract. In construction of an effort estimation model, it seems effective to use a window of training data so that the model is trained with only recent projects. Considering the chronological order of projects within the window, and weighting projects according to their order within the window, may also affect estimation accuracy. In this study, we examined the effects of weighted moving windows on effort estimation accuracy. We compared weighted and non-weighted moving windows under the same experimental settings. We confirmed that weighting methods significantly improved estimation accuracy in larger windows, though the methods also significantly worsened accuracy in smaller windows. This result contributes to understanding properties of moving windows.

1 Introduction

Software effort estimation is an important activity in software development. Its accuracy has a significant effect on project success. Research on the topic has studied two types of effort estimation approach: non-model-based methods (e.g. "expert judgment"), and model-based approaches (e.g. COCOMO, CART, etc.) [1]. A systematic review revealed that model-based software effort estimation models have been popular [2].

A software effort estimation model is developed from training data. Evaluation of the accuracy of the model is based on estimated efforts for testing data. Most studies split project data into training data and testing data randomly, or used a cross-validation approach.

In a practical sense, software projects can be ordered chronologically. Predicting the effort of future projects based on past projects, instead of forming training and testing sets, is more reasonable. Furthermore, it also seems appropriate to use recent projects as a basis of effort estimation. This is because old projects might be less representative of an organization's current practices.

Lokan and Mendes [3] examined whether using only recent projects improves estimation accuracy. They used a window to limit the size of training data so that an effort estimation model uses only recently finished projects. As new projects

J. Heidrich et al. (Eds.): PROFES 2013, LNCS 7983, pp. 214–228, 2013.

are completed, old projects drop out of the window. They found that estimation accuracy could increase by using the window.

Their view of a moving window assumes that old projects that are no longer in the window are not included have no value as training data, and projects within the window all have the same weight as training data. This does not take into account the chronological order of projects within the window. Projects within the window could be given different weights, according to their relative age to a target project. Weighting projects according to the order within a window may also affect estimation accuracy.

This study explored the effects of weighted moving windows for software effort estimation. The weighted moving windows generalizes the original moving windows. Recent projects receive higher importance than older projects. Linear regression models can consider different importance with *case weights*. The case weights can make a window have gradual weights.

In this paper, we addressed the following questions:

RQ1. Is there a difference in the accuracy of estimates between moving windows and weighted moving windows?

RQ2. If there is a difference, are there any insights with regards to trends with the use of different weighting functions?

2 Related Work

Research in software effort estimation models has a long history. However, few software effort estimation models were evaluated with consideration of the chronological order of projects.

Auer and Biffl [4] evaluated dimension weighting for analogy-based effort estimation, considering the effect of a growing data set. However, the authors used datasets having no date information. Thus, this evaluation method did not consider chronological order. Mendes and Lokan [5] compared estimates based on a growing portfolio with estimates based on leave-one-out cross-validation, using two different data sets. In both cases, cross-validation estimates showed significantly superior accuracy.

Some studies such as [6,7] used a project year in software effort estimation model construction. However, these studies did not consider chronological order in evaluation. Maxwell [8] demonstrated the construction and evaluation of software estimation model with the consideration of chronology. A candidate effort estimation model selected a year predictor. She also separated project data into training and test data according to a year.

To the best of our knowledge, Kitchenham et al. [9] first mentioned the use of moving windows. As a result of an experiment, they argued that old projects should be removed from the data set as new ones were added, so that the size of the dataset remained constant.

MacDonell and Shepperd [10] investigated moving windows as part of a study of how well data from prior phases in a project could be used to estimate later phases. They found that accuracy was better when a moving window of the 5

most recent projects was used as training data, rather than using all completed projects as training data.

Lokan and Mendes [3] studied the use of moving windows with linear regression models and a single-company dataset from the ISBSG repository. Training sets were defined to be the N most recently completed projects. They found the following insights: the use of a window could affect accuracy significantly; predictive accuracy was better with larger windows; some window sizes were 'sweet spots'.

Later they also investigated the effect on accuracy when using moving windows of various durations to form training sets on which to base effort estimates [11]. They showed that the use of windows based on duration can affect the accuracy of estimates, but to a lesser extent than windows based on a fixed number of projects.

This study is similar to [3] in that the same data set is investigated, using the same range of window sizes. It differs in that one additional independent variable is considered here, and models were based on variables selected with Lasso[12] instead of stepwise regression. In addition, this study differs from [3], and all other previous studies, by investigating different weights for projects of different ages: the main point of the paper.

3 Research Method

3.1 Dataset Description

The data set used in this paper is the same one analyzed in [3]. This data set is sourced from Release 10 of the ISBSG Repository. Release 10 contains data for 4106 projects; however, not all projects provided the chronological data we needed (i.e. known duration and completion date, from which we could calculate start date), and those that did varied in data quality and definitions. To form a data set in which all projects provided the necessary data for size, effort and chronology, defined size and effort similarly, and had high quality data, we removed projects according to the following criteria:

- The projects are rated by ISBSG as a high data quality (A or B).
- Implementation date and overall project elapsed time are known.
- Size is measured in IFPUG 4.0 or later (because size measured with an older version is not directly comparable with size measured with IFPUG version 4.0 or later). We also removed projects that measured size with an unspecified version of function points, and whose completion pre-dated IFPUG version 4.0.
- The size in unadjusted function points is known.
- Development team effort (resource level 1) is known. Our analysis used only the development team's effort.
- Normalized effort and recorded effort are equivalent. This should mean that the reported effort is the actual effort across the whole life cycle.
- The projects are not web projects.

Table 1. Summary statistics for ratio-scaled variables

Variable	Mean	Median	StDev	Min	Max
Size	496	266	699	10	6294
Effort	4553	2408	6212	62	57749
PDR	16.47	8.75	31.42	0.53	387.10

In the remaining set of 909 projects, 231 were all from the same organization and 678 were from other organizations. We only selected the 231 projects from the single organization, as the use of single-company data was more suitable to answer our research questions than using cross-company data. Preliminary analysis showed that three projects were extremely influential and invariably removed from model building, so they were removed from the set. The final set contained 228 projects.

We do not know the identity of the organization that developed these projects.

Release 10 of the ISBSG database provides data on numerous variables; however, this number was reduced to a small set that we have found in past analyses with this dataset to have an impact on effort, and which did not suffer from a large number of missing data values. The remaining variables were size (measured in unadjusted function points), effort (hours), and four categorical variables: development type (new development, re-development, enhancement), primary language type (3GL, 4GL), platform (mainframe, midrange, PC, multi-platform), and industry sector (banking, insurance, manufacturing, other).

Table 1 shows summary statistics for size (measured in unadjusted function points), effort, and project delivery rate(PDR). PDR is calculated as effort divided by size; high project delivery rates indicate low productivity. In [3], the authors examined the project delivery rate and found it changes across time. This finding supports the use of a window.

The projects were developed for a variety of industry sectors, where insurance, banking and manufacturing were the most common. Start dates range from 1994 to 2002, but only 9 started before 1998. 3GLs are used by 86% of projects; mainframes account for 40%, and multi-platform for 55%; these percentages for language and platform vary little from year to year. There is a trend over time towards more enhancement projects and fewer new developments. Enhancement projects tend to be smaller than new development, so there is a corresponding trend towards lower size and effort.

There are two ways in which a window size might be defined: by the number of projects [3], or duration [11]. A window based on duration can be scaled to include only those projects that reflect recent development projects and practices. In contrast, a window based on the number of projects can be scaled to provide enough data for sound analysis. This study defines a window as containing a fixed number of projects.

We adopted the same range of window sizes as [3]. The smallest window size was based on the statistical significance of linear regression with windowed project data: the smallest window size with which all regression models

Table 2. Formulae of weighted functions

Name	Formula
Triangular	$W(x) = 1 - \|x\|, \|x\| < 1$
Epanechnikov	$W(x) = 1 - x^2, \|x\| < 1$
Gaussian	$W(x) = \exp(-(2.5x)^2/2)$
Rectangular (Uniform)	$W(x) = 1, \|x\| < 1$

were statistically significant was 20 projects. The largest window size was based on the necessary number of testing projects for evaluation. As a result, we used window sizes from 20 to 120.

3.2 Weighted Moving Windows with Linear Regression

Linear regression is one of the popular methods for effort estimation. A typical effort estimation model is as follows:

$$\text{Effort} = b_0 + b_1 \text{Size} + \epsilon. \tag{1}$$

Here, b_0 and b_1 are regression coefficients, and ϵ represents an error term following a normal distribution. The regression coefficients are inferred from a training set so as to minimize the following function:

$$\sum_{i=1}^{n} \left(\text{Effort}_i - b_0 - b_1 \text{Size}_i\right)^2. \tag{2}$$

Here, n denotes the sample size of training set.

Equation 2 assumes that the errors of the training set are to be minimized equivalently. Weighted linear regression controls the importance of training projects via weighting. It minimizes the following function:

$$\sum_{i=1}^{n} w_i \left(\text{Effort}_i - b_0 - b_1 \text{Size}_i\right)^2. \tag{3}$$

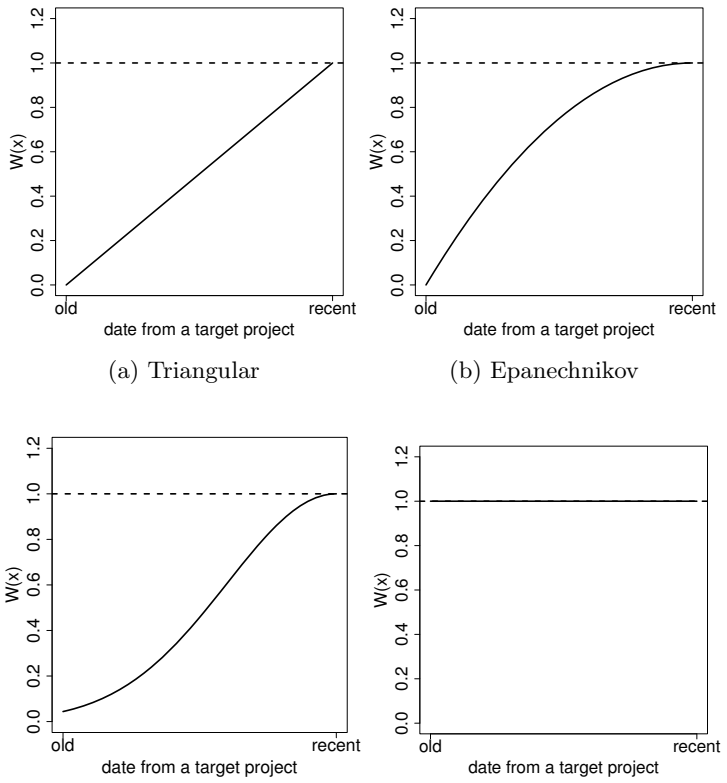
Here, w_i represents case weights for the training set.

From this perspective, an unweighted moving window assigns zero-weights to old projects, and equal weights to projects in the window. This formulation also reveals a point that the past study overlooked: it takes into account the chronological order of projects in the window.

This study weights projects in the training set so that a more recent project has a heavier weight. Table 2 shows four weight functions that we examined. We determined x as follows:

$$x = \frac{n_i}{n}. \tag{4}$$

Here, n_i represents a rank of project i in ascending order of date. That is, an older project takes a lower rank and x becomes smaller.



(a) Triangular (b) Epanechnikov

(c) Gaussian (d) Recutangular

Fig. 1. Weighted function forms

Figure 1 shows the forms of weighted functions. A rectangular function is equivalent to non-weighted moving windows. Different curve functions affect estimation accuracy differently. This study adopted three typical curves: linear, concave, S-shape. These functions are common in local regression [13].

3.3 Modeling Techniques

Weighted linear regression models were built using almost the same procedure as [3]:

1. The first step in building every regression model is to ensure numerical variables are normally distributed. We used the Shapiro-Wilk test on the training set to check if Effort and Size were normally distributed. Statistical significance was set at $\alpha = 0.05$. In every case, Size and Effort were not normally distributed. Therefore, we transformed them to a natural logarithmic scale.

2. Independent variables whose value is missing in a target project were not considered for inclusion in the estimation model.
3. Every model included Size as an independent variable. Beyond that, given a training set of N projects, no model was investigated if it involved more than $N/10$ independent variables (rounded to the nearest integer), assuming that at least 10 projects per independent variable is desirable [14].
4. Models were based on variables selected with Lasso[12] instead of stepwise regression because preliminary investigation showed that Lasso gave more accurate estimates than stepwise; details not presented here (Lasso implementation we used is the "`glmnet`" function from `glmnet` package for R.)
5. To verify the stability of an effort model, we used the following approach: Calculate Cook's distance values for all projects to identify influential data points. Any projects with distances higher than $(3 \times 4/N)$, where N represents the total number of projects, were removed from the analysis.

This procedure performs variable selection, and thus all variables introduced in Section 3.1 are just candidates for independent variables. Models constructed in our experiment can be different for every project.

3.4 Effort Estimation on Chronologically-Ordered Projects

This study evaluated the effects of moving windows of several sizes along with a timeline of projects' history. The effects were measured by performance comparisons beteen moving windows and a growing portfolio. A growing portfolio uses all past projects as the training set. No project ever gets a weight of zero. This evaluation method was performed with the following steps:

1. Sort all projects by starting time
2. Find the earliest project p_0 for which at least $w + 1$ projects, where w is a window size, were completed prior to the start of p_0
3. For every project p_i in chronological sequence, starting from p_0, form four estimates using weighted and non-weighted moving windows, and another using growing portfolio. For moving windows, the training set is the w most recent projects that finished before the start of p_i. For growing portfolio, the training set is all of the projects that finished before the start of p_i.
4. Evaluate estimation results.

3.5 Performance Measures

Performance measures for effort estimation models are based on the difference between estimated effort and actual effort. As in previous studies, this study used MMRE, PRED(25), and MMAE [1] for performance evaluation.

To test for statistically significant differences between accuracy measures, we used the Wilcoxon ranked sign test and set statistical significance level at $\alpha = 0.05$. `wilcoxsign_test` function of `coin` package for R was used with default options. We also controlled false dicovery rate (FDR) of multiple testing [15] with the `qvalue` function of `qvalue` package. FDR is a ratio of the number of falsely rejected null hypotheses to the number of rejected null hypotheses.

Table 3. Mean absolute residuals with different window sizes

Window size (N)	Testing Projects	Growing	(a) MAE	(a) p–val.	(b) MAE	(b) p–val.	(c) MAE	(c) p–val.	(d) MAE	(d) p–val.
20	201	2638	2830	0.107	2829	0.023	2879	0.087	2709	0.140
30	178	2578	2613	0.894	2552	0.917	2648	0.849	2570	0.651
40	165	2541	2599	0.358	2520	0.716	2618	0.342	2523	0.834
50	153	2527	2483	0.973	2423	0.616	2452	0.893	2376	0.460
60	136	2458	2279	0.204	2268	0.302	2262	0.246	2320	0.101
70	126	2300	2147	0.054	2178	0.222	2070	0.087	2060	0.122
80	126	2300	2083	0.011	2033	0.001	2111	0.015	2239	0.084
90	111	2236	2022	0.004	1967	0.001	2074	0.057	2053	0.001
100	88	2314	1930	0.002	2041	0.002	2051	0.041	2192	0.001
110	75	1981	1684	0.004	1771	0.011	1662	0.012	1845	0.000
120	71	1982	1715	0.002	1782	0.006	1696	0.002	1852	0.002

(a) Triangular, (b) Epanechnikov, (c) Gaussian, (d) Rectangular

4 Results

4.1 Accuracy with Different Window Sizes

Tables 3 and 4 show the effects of window sizes on mean absolute residuals and mean MRE. The first column shows window sizes, and the second column shows the total number of projects used as a target project with the corresponding window size. The larger a window size is, the smaller the number of testing projects is. The 3rd column shows accuracy measures with a growing portfolio for the corresponding window sizes. The 4th column shows accuracy measures for the Triangular function. The 5th column shows the p–value from statistical tests on accuracy measures between a growing portfolio and the Triangular function. The remained columns show accuracy measures and p–value for the other weighted functions. The results were computed for all window sizes; the tables only show every tenth window size, due to space limitations. This is still sufficient to show the essential trends.

Figures 2 and 3 show the difference in mean MAE and mean MRE between a growing portfolio and moving windows. The x-axis is the size of the window, and the y-axis is the subtraction of the accuracy measure value with a growing portfolio from that with moving windows at the given x-value. Smaller values of MAE and MRE are better, so the window is advantageous where the line is below 0. Circle points mean a statistically significant difference, in favor of moving windows. Square points mean a statistically significant difference, with moving windows being worse than a growing portfolio. We consider a window is effective if the corresponding q-value is below 0.05 (this means that that the number of falsely rejected hypotheses was at most 5% of rejected hypotheses).

Figures and tables revealed common characteristics among weighted and non-weighted moving windows:

Table 4. Mean MRE with different window sizes

Window size (N)	Testing Projects	Growing	(a) MRE	(a) p–val.	(b) MRE	(b) p–val.	(c) MRE	(c) p–val.	(d) MRE	(d) p–val.
20	201	1.28	1.47	0.171	1.56	0.021	1.53	0.118	1.45	0.100
30	178	1.35	1.45	0.839	1.37	0.664	1.49	0.927	1.26	0.612
40	165	1.35	1.43	0.539	1.37	0.856	1.45	0.426	1.31	0.914
50	153	1.39	1.35	0.917	1.26	0.630	1.39	0.852	1.27	0.316
60	136	1.42	1.23	0.213	1.23	0.364	1.25	0.219	1.23	0.027
70	126	1.48	1.34	0.038	1.37	0.061	1.24	0.037	1.31	0.020
80	126	1.48	1.29	0.000	1.29	0.000	1.29	0.001	1.36	0.001
90	111	1.37	1.20	0.000	1.17	0.000	1.24	0.002	1.17	0.000
100	88	1.36	1.09	0.000	1.13	0.000	1.16	0.007	1.18	0.000
110	75	1.39	1.12	0.000	1.15	0.000	1.14	0.000	1.16	0.000
120	71	1.38	1.12	0.000	1.13	0.000	1.08	0.000	1.18	0.001

(a) Triangular, (b) Epanechnikov, (c) Gaussian, (d) Rectangular

- With smaller windows, all measures are always better using a growing portfolio. Shown in Figs. 2(b) and 3(b), the difference was significant only in one window size. For mean MRE, the difference was found only in the case of using Epanechnikov function.
- In medium windows, moving windows become advantageous. The window size where it becomes advantageous is different among types of weighted functions. Using the Rectangular function becomes advantageous at a smaller window size than using the other functions.
- With larger windows, all measures are always better using moving windows, and the difference was significant in $70 \leq w \leq 120$. Some weighted functions showed significance in smaller windows. Improvements in mean MRE range from 10% to 30%, averaging 20%. Improvements in mean MAE range from 70 to 384, averaging 231. The difference was larger when using gradually weighted functions.

The difference in PRED(25) also showed similar trends, though the results are not shown due to space limitations. With smaller windows, a growing portfolio showed better performance than moving windows. With larger windows, moving windows became advantageous.

4.2 Accuracy Comparisons among Different Weighting Functions

Figures 4 and 5 show the difference in mean MAE and mean MRE between Rectangular and the other functions. Weighted moving windows is advantageous where the line is below 0. Figures 4 and 5 reveal the following:

- With smaller windows, using the Rectangular function is advantageous. The difference was significant around $20 \leq w \leq 50$ for MMRE.
- With medium windows, weighted and non-weighted moving windows are competitive. There was no clear preference between them. There was no significant difference.

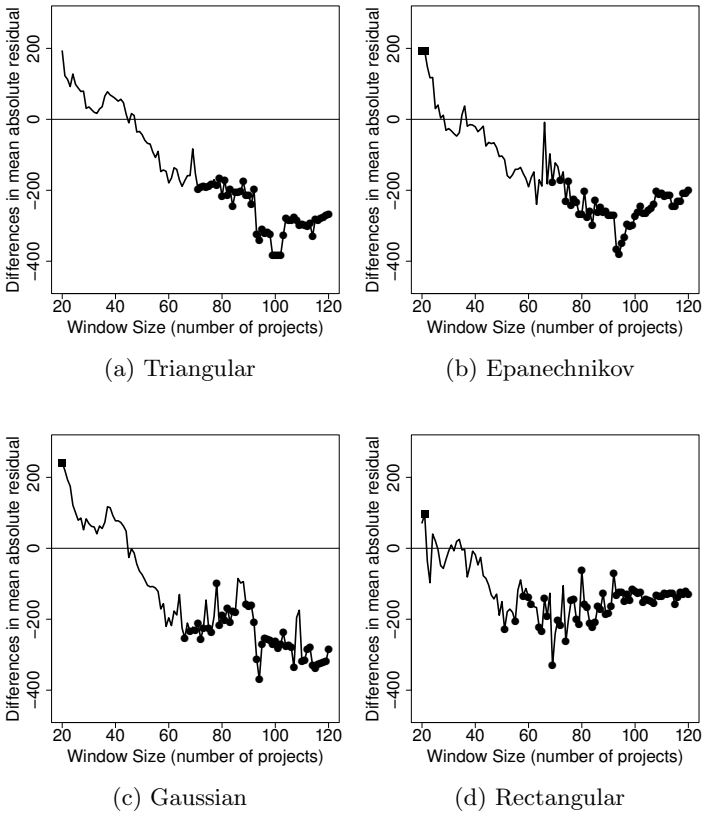
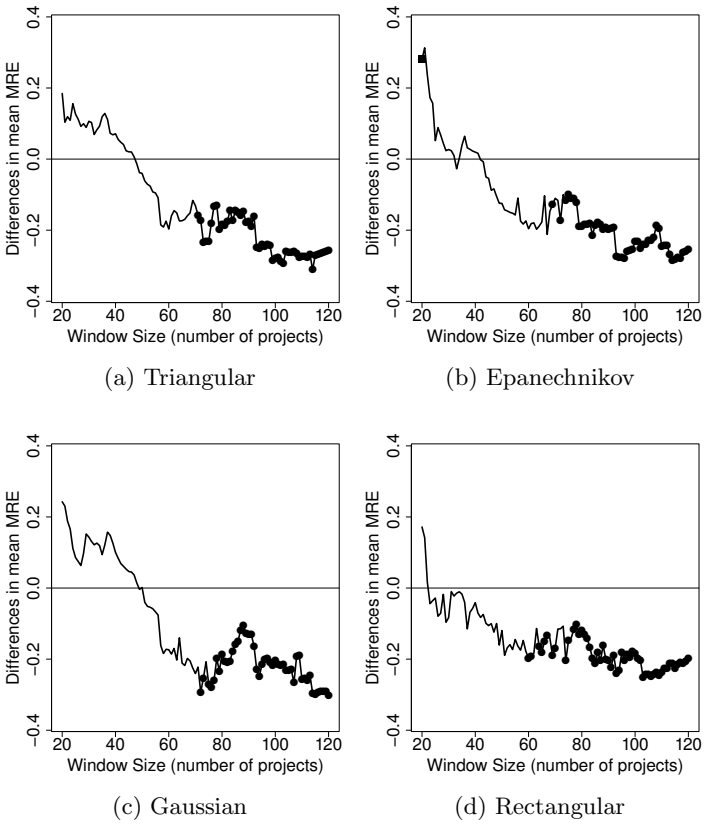


Fig. 2. The difference of accuracy measures between growing and windowing (mean MAE)

– With larger windows, weighted moving windows is advantageous. The range of advantageous window sizes were different among types of weighted functions. The lines in Fig. 4 are always below zero when $w > 90$ for the three functions. The lines in Fig. 5 sometimes rise above zero for Epanechnikov and Gaussian functions. However, statistical tests supported only the weighted functions. For statistically significant windows, improvements in mean MRE range from 1% to 10%, averaging 5%. No significance in case of mean MAE.

The small number of rejected hypotheses might cause the insignificance for mean MAE. With the small number of rejected hypotheses, for instance 10, q-value of 0.05 would limit the number of falsely rejected hypotheses to $0.05 \times 10 < 1$. We expect any of significant windows and can allow looser q-value. With q-value of 0.2, for instance, we could also find significances for mean MAE. Thus, we could say weighted moving windows also had posivitve effects on mean MAE.



(a) Triangular (b) Epanechnikov

(c) Gaussian (d) Rectangular

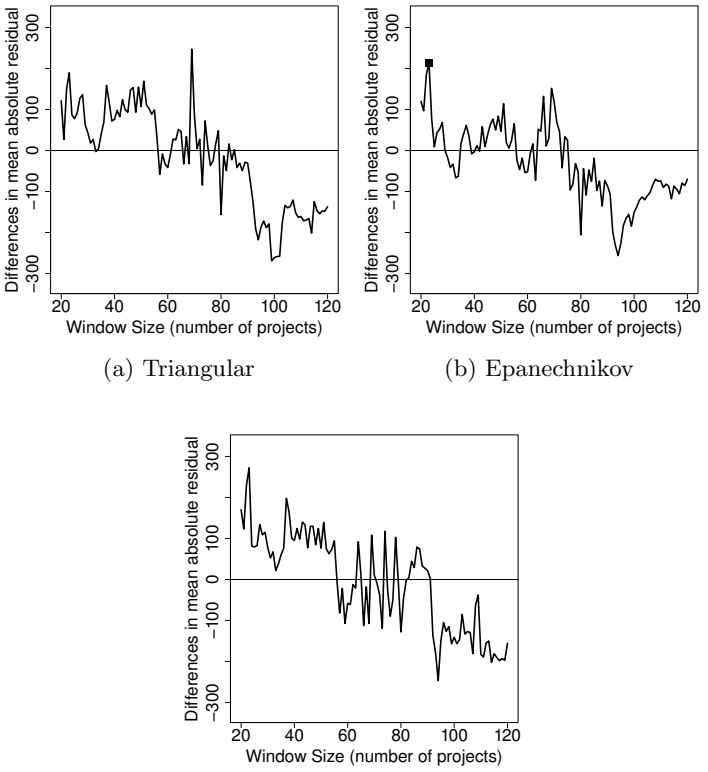
Fig. 3. The difference of accuracy measures between growing and windowing (mean MRE)

5 Discussion

5.1 Answer to RQ1

The null hypothesis was rejected on statistical tests for the difference between all weighted moving windows and a growing portfolio. For Epanechnikov function, for instance, the null hypothesis was rejected in window duration of 20 and 21, and from 69 to 120, based on mean MAE; 21 and from 69 to 120, based on mean MRE. The use of weighted moving windows can affect estimation accuracy against a growing portfolio.

On statistical tests for the difference between weighted and non-weighted moving windows, the null hypothesis was also rejected. For the Triangular function, for instance, the null hypothesis was rejected in window duration from 22 to 51, and from 99 to 120, based on mean MRE. The difference based on mean MAE



(a) Triangular

(b) Epanechnikov

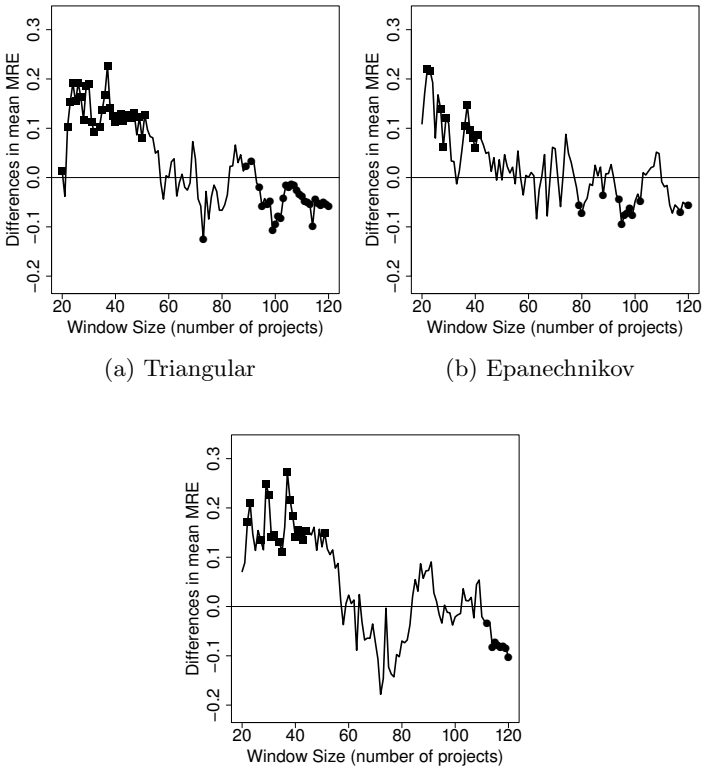
(c) Gaussian

Fig. 4. The difference of accuracy measures between Rectangular and the other weighted functions (Mean MAE)

was insignificant through all window sizes. However, Fig. 4 depicted positive effects of moving window approach. We thus concluded that the use of weighted moving windows can also affect estimation accuracy against non-weighted moving windows.

5.2 Answer to RQ2

Weighted moving windows showed inferior estimation accuracy when using small windows. The difference was statistically significant, and there is an advantage in using non-weighted moving windows. However, weighted functions perform as well as a growing portfolio: the difference of estimation accuracy between them was insignificant. In contrast, weighted moving windows showed superior estimation accuracy when using larger windows. The difference was also statistically significant, and there is an advantage in using moving windows.



(a) Triangular

(b) Epanechnikov

(c) Gaussian

Fig. 5. The difference of accuracy measures between Rectangular and the other weighted functions (Mean MRE)

The above characteristics can be explained with an interaction between window sizes and the steepness of weighted function curves. The Triangular, Epanechnikov and Gaussian functions assign a small (near to zero) weight to the oldest projects in a window. The other projects receive heavier weights in accordance with distance from the date of a target project within the window. With small size windows, weighted function assigns steeply declining weights. With large window sizes, a weighted function assigns gently declining weights. When the degree of steepness meshes with a window size, a weighted function contributes to improvement of estimation accuracy.

The difference in advantageous window sizes among weighted functions supports the explanation. Figure 1 depicts the difference of steepness among weighted functions. Gaussian is the steepest function, and Epanechnikov is the most gentle function. The steepness of Triangular function is between them. Non-weighted moving windows assigns equal weights and is more gentle than Epanechnikov

function. In Figs. 2 and 3, steep functions became advantageous more slowly than gentle functions. Those functions are too steep to reflect the importance of recent projects. With large windows, steep functions meshed with window sizes and improved estimation accuracy significantly. However, the range of significant windows was narrower than that of gentle functions as shown in Figs. 4 and 5.

The results support that weighted moving windows can improve estimation accuracy with appropriate steepness.

6 Threats to Validity

This study shares the same threats to validity as the previous studies.

First, we used only one dataset. The dataset is a convenience sample and may not be representative of software projects in general. Thus, the results may not be generalized beyond the dataset; this is true of all studies based on convenience samples. We trust that numerous potential sources of variation can be removed from the dataset by the selection of a single-company dataset. Since the dataset is large and covers a long time span, we assume it is a fair representation of this organization's projects. The inclusion of the sector as an independent variable helps to allow for variations among sectors in the dataset.

Second, all the models employed in this study were built automatically. Automating the process necessarily involved making some assumptions, and the validity of our results depends on those assumptions being reasonable. For example, logarithmic transformation is assumed to be adequate to transform numeric data to an approximately normal distribution; residuals are assumed to be random and normally distributed without that being actually checked; multi-collinearity between independent variables is assumed to be handled automatically by the nature of Lasso. Based on our past experience building models manually, we believe that these assumptions are acceptable. One would not want to base important decisions on a single model built automatically, without at least doing some serious manual checking, but for calculations such as chronological estimation across a substantial data set we believe that the process here is reasonable.

Third, this study used weighted linear regression. Many effort estimation models have been proposed, and each model can show better accuracy in particular situations. However, linear regression is a popular and accurate effort estimation models. We thus think it is a good choice among major effort estimation models.

7 Conclusion

This paper investigated the use of weighted moving windows as a way to improve non-weighted moving windows. We have shown that it has a statistically significant effect on estimation accuracy in terms of MRE. Although different weighted functions affected estimation accuracy differently, weighted moving windows were significantly advantageous in larger windows. Non-weighted moving windows were significantly advantageous with smaller windows.

What these results suggest is that it can be better to use a weighted window of projects with a weighted function having appropriate steepness. Weighted moving windows gradually decrease the importance of past projects. If a decrease curve is too steep or too gentle, weighted moving windows makes estimation accuracy worse. How to determine appropriate steepness is a crucial question

Our future work involves replication with other datasets such as Maxwell dataset and the CSC dataset used in past studies.

References

1. Port, D., Korte, M.: Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research. In: Proc. of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM (2008)
2. Jørgensen, M., Shepperd, M.: A Systematic Review of Software Development Cost Estimation Studies. IEEE Trans. Softw. Eng. 33(1), 33–53 (2007)
3. Lokan, C., Mendes, E.: Applying moving windows to software effort estimation. In: Proc. of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 111–122 (2009)
4. Auer, M., Biffl, S.: Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting. In: Proc. of International Symposium on Empirical Software Engineering, pp. 147–155. IEEE (2004)
5. Mendes, E., Lokan, C.: Investigating the use of chronological splitting to compare software cross-company and single-company effort predictions: a replicated study. In: Proc. of the 13th Conference on Evaluation & Assessment in Software Engineering (EASE 2009). BCS (2009)
6. Keung, J.W., Kitchenham, B.A., Jeffery, D.R.: Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation. IEEE Trans. Softw. Eng. 34(4), 471–484 (2008)
7. Li, J., Ruhe, G.: Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+. Empir. Softw. Eng. 13(1), 63–96 (2007)
8. Maxwell, K.D.: Applied Statistics for Software Managers. Prentice Hall (2002)
9. Kitchenham, B., Lawrence Pfleeger, S., McColl, B., Eagan, S.: An empirical study of maintenance and development estimation accuracy. J. Syst. Softw. 64(1), 57–77 (2002)
10. MacDonell, S.G., Shepperd, M.: Data accumulation and software effort prediction. In: Proc. of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM (2010)
11. Lokan, C., Mendes, E.: Investigating the Use of Duration-based Moving Windows to Improve Software Effort Prediction. In: Proc. of the 19th Asia-Pacific Software Engineering Conference, pp. 819–927. IEEE Computer Society (2012)
12. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Statist. Soc. Ser. B, 267–288 (1996)
13. Loader, C.: Local Regression and Likelihood. Statistics and Computing. Springer
14. Tabachnick, B.G., Fidell, L.S.: Using Multivariate Statistics. Harper-Collins (1996)
15. Storey, J.D.: A direct approach to false discovery rates. J. Roy. Statist. Soc. Ser. B 64, 479–498 (2002)