Abstract

Shepperd and MacDonell “Evaluating prediction systems in software project estimation”. Information and Software Technology 54 (8), 820–827, 2012, proposed an improved measure of the effectiveness of predictors based on comparing them with random guessing. They suggest estimating the performance of random guessing using a Monte Carlo scheme which unfortunately excludes some correct guesses. This biases their MARP0 to be slightly too big, which in turn causes their standardised accuracy measure SA to over estimate slightly. In commonly used software engineering datasets it is practical to calculate an unbiased MARP0 exactly.

1. Introduction

Shepperd and MacDonell recently reported problems with often used measures of performance prediction used in software engineering [Shepperd and MacDonell, 2012]. In particular they report mean magnitude relative error MMRE “will be biased towards prediction systems that under-estimate” (Shepperd and MacDonell, 2012, page 822) and so they recommend MMRE not be used. Instead they propose standardised accuracy measure SA be used instead:

\[
\text{Standardised Accuracy} = \text{SA} = 1 - \frac{\text{MAR}}{\text{MARP}_0} \times 100
\]

Where MAR is the mean of the absolute error for the predictor of interest. E.g. for software project estimation, the average of the absolute difference between the effort predicted and the actual effort the project took. To allow easy comparison they normalise MAR by dividing it by the same measure for random guessing (MARP0). They suggest calculating MARP0 by taking the average error of 1000 runs of random guessing. Instead we show (Equation 1 page 18) the correct exact value can be calculated immediately. Nevertheless Figure 2 makes clear typically the average of 1000 runs converges to be very close to the long term average. They define random guessing prediction as returning one of the other actual measurements of project effort chosen at random. Notice they do not consider that random prediction might stumble across the correct answer by chance. This increases their estimate of the average error. Therefore their Monte Carlo estimate of MARP0 converges towards a value higher than it should be. The difference is small, their MARP0 is on average \(n/(n - 1)\) times higher than it should be. In the case of their Atkinson-2 data set their MARP0 will be on average 7% higher than it should be. For this particular dataset and prediction technique, correcting the bias in MARP0 would lead to the predictor’s standardised accuracy (SA) being about 7% lower.

As they define MARP0, as a result of random sampling, there will always be some variation due to noise. To avoid this complicating subsequent analysis their random MARP0 should be estimated only once per dataset.

Unfortunately it is common in software engineering prediction experiments to have only a few datasets [Shepperd and MacDonell, 2012] some of which may be quite small. For most software engineering prediction datasets it is feasible to calculate exactly the average absolute error for such a random guess predictor. This is because for data sets with \(n\) outcomes the exact value of MARP0 can be found in \(O(n^2)\) steps. Indeed if there are 2000 or fewer results (e.g. software projects) in the dataset it is faster (needs fewer residuals) to calculate MARP0 exactly, rather than use a Monte Carlo estimate as suggested by (Shepperd and MacDonell, 2012, page 822). For the Atkinson-2 data set, Figure 2 shows only 120 calculations of abs(measurement vs. random prediction) are needed to get the exact value (263.508), rather than 1000 runs (potentially each uses \(n\) abs() steps) to get a value 7% bigger than it should be.
Figure 1: Histogram of 1000 MAR$_P$ values from naive guessing of the Atkinson-2 (Shepperd and Cartwright, 2005) Table 11 data set. (New random sample.) Notice Shepperd and MacDonell (2012) tends to suggest random guessing (sample mean 281.2) performs worse than it actually does (exact mean 263.508). The variance is 2118 (SD 46) giving a standard error of 1.46 (plotted as the error bar in Figure 2).

Figure 2: Only $n(n-1)/2$ ($n$ is size of dataset) calculations are needed to find exact mean absolute error for random guessing. Whereas Shepperd and MacDonell (2012) recommend 1000 samples (each of $n$ calculations), which converges to an overestimate of MAR$_P$. (Same data as Figure 1.) Error bar shows expected variation (i.e. standard error) after 1000 samples.
2. Conclusions

The mean absolute error of a predictor which randomly guesses is essential for the normalisation of the standardised accuracy measure SA proposed by Shepperd and MacDonell (2012). However the calculation they proposed is stochastic and biased. The exact calculation

\[
\text{MAR}_{P_0} = \frac{2}{n^2} \sum_{i=1}^{n} \sum_{j<i} \left| y_i - y_j \right|
\]

avoids the bias by allowing random guesses to alight on the correct answer (i.e., \(|\cdot| = 0\) when \(i = j\)) and is actually faster (i.e. fewer \(|y_i - y_j|\) residuals are needed) than a naive Monte Carlo estimate of \(\text{MAR}_{P_0}\) when \(n(n-1)/2 < 1000n\), i.e. when \(n \leq 2000\). Since the calculation is exact, there are no issues associated with stochastic variations.

Except for large data sets, when calculating SA, the unbiased exact version of \(\text{MAR}_{P_0}\) should usually be used.

Acknowledgements

I would like to thank Martin Shepperd for helpful suggestions and encouragement.

Appendix A. Precalculated \(\text{MAR}_{P_0}\) for Popular Datasets

Although it is straight forward to calculate \(\text{MAR}_{P_0}\) with Equation 1, in Table A.1 we provide \(\text{MAR}_{P_0}\) for some commonly used software engineering prediction benchmarks (see also gawk script in exact_marp0.tar).

Table A.1: Mean absolute error of random prediction \(\text{MAR}_{P_0}\) [Shepperd and MacDonell (2012)] for the latest version (Nov 2015) of the PROMISE public software engineering effort estimation benchmarks. Note column 6 (MAR\(_{P_0}\)) depends upon the units used and is not strictly comparable across datasets.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>records</th>
<th>attributes</th>
<th>predicted attribute</th>
<th>units</th>
<th>MAR of random prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>kitchenham</td>
<td>145</td>
<td>10</td>
<td>Actual.effort</td>
<td></td>
<td>3771.66</td>
</tr>
<tr>
<td>kitchenham</td>
<td>132</td>
<td>10</td>
<td>Actual.effort (exclude missing data)</td>
<td></td>
<td>3961.26</td>
</tr>
<tr>
<td>miyazaki94</td>
<td>48</td>
<td>9</td>
<td>MM</td>
<td>Man-Months</td>
<td>111.465</td>
</tr>
<tr>
<td>kemerer</td>
<td>15</td>
<td>8</td>
<td>EffortMM</td>
<td></td>
<td>209.498</td>
</tr>
<tr>
<td>china</td>
<td>499</td>
<td>19</td>
<td>Effort</td>
<td></td>
<td>4915.13</td>
</tr>
<tr>
<td>Albrecht</td>
<td>24</td>
<td>8</td>
<td>Effort</td>
<td></td>
<td>24.3396</td>
</tr>
<tr>
<td>maxwell</td>
<td>62</td>
<td>27</td>
<td>Effort</td>
<td></td>
<td>8661.64</td>
</tr>
<tr>
<td>cocomo-sdr</td>
<td>12</td>
<td>25</td>
<td>ACTUAL EFFORT</td>
<td>man month</td>
<td>5.86667</td>
</tr>
<tr>
<td>nas93</td>
<td>93</td>
<td>24</td>
<td>act_effort</td>
<td>(one month=152 hours)</td>
<td>840.433</td>
</tr>
<tr>
<td>coc81</td>
<td>63</td>
<td>19</td>
<td>actual</td>
<td></td>
<td>1100.47</td>
</tr>
</tbody>
</table>

We downloaded\(^1\) all the public software effort estimation datasets from tera-Promise. (PROMISE is one of the largest repositories specialising in software engineering research datasets Menzies et al. (2015)).

There are a variety of ways of handling missing data. Two common approaches are 1) supply defaults for or ignore the missing attributes and 2) exclude cases where one or more data are unknown. Accordingly in Table A.1 (the two kitchenham rows) we provide \(\text{MAR}_{P_0}\) for both approaches (notice difference in number of records).

References


\(^1\)http://openscience.us/repo/effort/