Noname manuscript No.
(will be inserted by the editor)

Kernel Methods for Software Effort Estimation

Effects of different kernel functions and bandwidths on estimation accuracy

Received: date / Accepted: date

Abstract Analogy based estimation (ABE) generates an effort estimate for a new software project through adaptation of similar past projects (a.k.a. analogies). Majority of ABE methods follow uniform weighting in adaptation procedure. In this research we investigated non-uniform weighting through kernel density estimation. After an extensive experimentation of 19 datasets, 3 evaluation criteria, 5 kernels, 5 bandwidth values and a total of 2090 ABE variants, we found that: 1) non-uniform weighting through kernel methods cannot outperform uniform weighting ABE and 2) kernel type and bandwidth parameters do not produce a definite effect on estimation performance. Hence, -provided that similar experimental settings are adopted- we discourage the use of kernel methods as a weighting strategy in ABE.

Keywords Effort estimation, data mining, kernel function, bandwidth

1 Introduction

If a researcher or an industrial practitioner reads the literature on software effort estimation, they will encounter a dauntingly large number of different estimation methods. For example, the following is a partial list of some of the methods currently being used:

- Boehm uses linear regression [6].
- Shepperd prefers analogy-based methods [51].
- Auer et al. [3] uses extensive search to find weights for project features.
- Pendharkar et al. used Bayesian Network (BN) for effort estimation and incorporated BN into decision making procedure against risks [44].
- Mendes and Mosley used a data-driven Bayes net for web effort estimation [34].
- Li et al. combine feature weighting with instance selection [32].

This list is hardly complete. Elsewhere [28, 38], we have studied all the different kinds of *Analogy-Based Estimation* (hereafter, ABE) methods in the literature:

- ABE generates estimates by sampling the neighborhood of some test instance.

Address(es) of author(s) should be given

– The exact sampling method is controlled by the *kernel method* which we divide into *uniform* and *non-uniform* (here after U-ABE and N-ABE, respectively).
– Uniform methods treat all items in the local neighborhood in the same way.
– Non-uniform methods weight those neighbors in different ways.

Our list of ABE variants is presented in the next section: we can find over 17,000 different ways to configure ABE-style estimators. How we select the right method from this large menagerie of possibilities? One way is to try many options, then see what works best on the local data. Baker then Menzies et al., used exhaustive search through all project features, learners and other variables [4, 37]. The CPU intensive nature of that approach begs the question: is there a simpler way?

There is indirect evidence that there must be a simpler way. If we look at the size of the training data available for effort estimation, it is usually only a few dozen (or less) instances. For example:

– The data accessible to researchers in four recent publications [3, 4, 32, 36] have median size of 13, 15, 33, 52, respectively.
– Later in this paper we list the 19 data sets used in this study: they vary in size from 10 to 93 with a median of 28.

Given that the size of these data sets is so small, it seems unreasonable to believe that (e.g.) complex multi-dimensional partitioning schemes can find more than more simplistic methods such as "take the median of the variables in the local region".

This is indeed the case. The experiments of this paper show that, for ABE, the simplest kernel schemes are *most often as good as anything else*. This is an important result since it means that future researchers have less to explore (at least, in the field of ABE). Hopefully, if more researchers critically reviewed the space of options for their tools, then we will arrive at a much smaller and much more manageable set of candidate effort estimation methods.

The rest of the paper is organized as follows. To focus the paper, we will answer the following research questions:

RQ1 Is there evidence that non-uniform weighting improves the performance of ABE?
RQ2 What is the effect of different kernels for non-uniform weighting in ABE?
RQ3 What is the effect of different bandwidths?
RQ4 How do the characteristics of software effort datasets influence the performance of kernel weighting in N-ABE?

To do so, Section 2 summarizes our motivation of this story. Section 3 talks about the value of negative results. In Section 4 we provide background information regarding related work. Section 5 explains the adopted experimental methodology. In Section 6 results are presented. Section 7 is a discussion section and in Section 8 the threats to the validity of the results are presented. In Section 9 we summarize our conclusions and answer the research questions. Finally in Section 10 we list future directions of this research.

Note that, for reasons of space, our results will be presented in a summary format. For the full results, see http://goo.gl/qpQiD.

2 Motivation

This work is part of an on-going investigation into effort estimation. Effort estimation is important since, those estimates are often wrong by a factor of four [6] or even more [19].

As a result, the allocated funds may be inadequate to develop the required project. In the worst case, over-running projects are canceled and the entire development effort is wasted.

We study analogy-based estimation (ABE), for several reasons:

- It is a widely studied approach [3, 18, 20, 21, 23, 24, 29–32, 50, 51, 54].
- It works even if the domain data is sparse [42].
- Unlike other predictors, it makes no assumptions about data distributions or an underlying model.
- When the local data does not support standard algorithmic/parametric models like COCOMO, ABE can still be applied.

Based on a literature review of just one sub-section of the field [38], we have found at least eight dimensions that distinguish different ABE methods:

A : The distance measure used to compute similarity;
B : The "neighborhood" function that decides what is a "near" neighbor;
C : The method used to summarize the nearest neighbors;
D : The instance selection mechanism;
E : The feature weighting mechanism;
F : The method for handling numerics, e.g. logging, discretization, etc.

That review found in the literature three to nine variants of A, B, C, D, E, F (which combine to a total over 17,000 variants). Some of these variants can be ruled out, straight away. For example, for ABE that reasons only about the single nearest neighbor, then all the summarization mechanisms return the same result. Also, not all the feature weighting techniques require discretization, thereby further decreasing the space of options. However, even after discarding some combinations, there are still thousands of possibilities to explore.

In our view, it is unacceptable that researchers continually extend effort estimation methods without trying to prune away the less useful variants. To that end, in previous work, we have tried to rank and prune estimation methods based on model selection [38] or feature weighting [37] or instance selection [28].

We have had much recent success in pruning different variants:

- For COCOMO-style data [6], only four variants were demonstrably better than the another 154 variants [38].
- Also, in non-COCOMO data, we have found 13 variants that perform better than 77 others [22].

This paper is our first exploration of kernel methods. Kernel methods are important since they comment on many of the options within A, B, C, D, E, F listed above:

- Simpler kernel methods mean simple neighborhood and summarization methods.
- In theory, better estimates could be generated by a smarter sampling of the neighborhood. For example, an intelligent selection of the kernel might compensate for data scarcity.

We study kernel estimation since, if the effort data corresponds to a particular distribution, then it would seem wise to bias that sampling by that distribution. Also, at least one other research team in the field of effort estimation have also begun exploring different kinds of kernel methods [35, 36].

Despite the potential of kernel methods to improve effort estimation, it is an unexplored area. For the most part, researchers in this area propose kernel methods with minimal motivation or experimentation [8, 14, 35, 36]. Hence, prior to performing the experiments of this paper, we believed that kernel methods would be a rich source of future insights into effort estimation:

- The space of sampling and weighting schemes seen in the SE literature is much smaller than that seen in other fields (see for example the literature from data mining or signal processing [13, 15, 43]).
- Hence, it seemed to us that a rigorous exploration of this under-explored area might be a worthy topic of research, perhaps applying methods not yet used in the SE literature.

This paper presents that rigorous exploration and leads to the the negative result summarized in the conclusion (that simple kernel methods do as well as anything else, at least for ABE).

3 On the Value of Negative Results

While it would have been gratifying to have found a positive result (e.g. that some kernel method was very much better), it is important to report such negative results as well. A very thorough discussion on the value of negative results can be found in [9]. The fundamental question is whether a negative result poses a positive knowledge. Positive knowledge is defined by Browman et al. to be the ability of being certain, not being either right or wrong [9]. However, not all certain conclusions are *knowledge* per se. Common concerns are:

i. is the topic/hypothesis plausible,
ii. are the experiments sound,
iii. do the results propose *"negative evidence"* or *"non-conclusive search"* and
iv. will the reported results be valuable to future research.

As for ***i.***, research on weighting methods in ABE is quite plausible, see weighting method proposed earlier by Mendes et al. [35, 36]. In that respect, our evidence of negative results serve the purpose of guiding research away from conclusions (such as kernel weighting can improve ABE performance) that would otherwise seem reasonable [9].

When presenting negative evidence it is crucially important to have *sound and extensive* experimentation (condition ***ii.***). This report rigorously investigates kernel weighting on *19* datasets subject to *3* performance measures through appropriate statistical tests.

The idea behind condition ***iii.*** is that *"one should disvalue inconclusive results"* [9], i.e. negative conclusions are more meaningful than *uncertainty*. The kernel weighting experiments of this paper on a wide range of ABE variants are negative evidence to conclude that it does not improve ABE performance, thereby satisfying ***iii.*** Finally condition ***iv.*** questions the benefit of results to future research. After years of research, effort estimation still suffers from conclusion instability. For stable conclusions, retiring a considerable portion of search space is as important as the discovery of successful applications. The contribution of this work is through retirement of *2090* of ABE variants.

4 Background

4.1 Software Effort Estimation

We can divide software effort estimation into at least two groups [48]: *expert judgment* and *model-based* techniques. *Expert judgment* methods depend on consolidation of expert opinions regarding the cost of a new project and are widely used in software effort estimation practices [16]. Expert judgment can be applied either explicitly (following a method like Delphi [5]) or implicitly (informal meetings among experts). Unlike expert-based methods, *model-based techniques* do not rely heavily on human judgment. Model based techniques are products of:

1) Algorithmic and parametric approaches or
2) Induced prediction systems.

The first approach is the adaptation of an expert-proposed model to local data. A widely known example to such an approach is Boehm's COCOMO method [6]. The second approach is particularly useful in the case where local data does not conform to the specifications of the expert's method. A few examples of induced prediction systems are linear regression, neural nets, model trees and analogy based estimation [37, 49]. There are also successful applications where expert and model based techniques are integrated to complement one another [7, 27]. In particular when such estimation practices are employed iteratively over time, the estimation accuracy can significantly be improved. For example in [53], an integrated approach called CoBRA was applied in an iterative manner and accuracy was improved from 120% error down to 14%.

Analogy based estimation (ABE) or estimation by analogy (EBA) is a form of case based reasoning (CBR) and it is grouped together with induced prediction systems. ABE generates its estimate for a new project by gathering evidence from similar past projects. When we analyze the previous research of experts on the domain of ABE such as Shepperd et al. [51], Mendes et al. [36] and Li et al. [32], we can see a baseline technique lying under all ABE methodologies. The baseline technique is composed of the following steps:

- Form a table (training set) whose rows are completed past projects and whose columns are *independent* variables (the features that define projects) and a *dependent* variable (the recorded effort value).
- Decide on the number of similar projects (*analogies*) to use from the training set, i.e k-value.
- For each test instance, select k analogies out of the training set.
  - While selecting analogies, use a similarity measure (for example the Euclidean distance).
  - Before calculating similarity, apply a scaling measure on independent features to equalize their influence on this similarity measure.
  - Use a feature weighting scheme to reduce the effect of less informative features.
- Adapt the effort values of the k nearest analogies to come up with an effort estimate.

Following the steps of this baseline technique, we define a framework called ABE0. ABE0 uses the Euclidean distance as a similarity measure, whose formula is given in Equation 1. In Equation 1, w_i corresponds to feature weights applied to independent features. ABE0 framework does not favor any features over the others, therefore each feature has equal importance in ABE0, i.e. $w_i = 1$.

$$Distance = \sqrt{\sum_{i=1}^{n} w_i(x_i - y_i)^2} \tag{1}$$

The next step is deciding on how to adapt project costs. There is a wide variety of adaptation strategies in the literature [33]. Using effort value of the nearest neighbor [8], taking mean [41] or median [2] of closest analogies, inverse distance and inverse rank weighted mean of closest analogies are among the commonly used adaptation methods [33]. Angelis et al. suggest that as the number of the closest projects increase, median is a robust solution [2]. They have found that taking median instead of mean decreases the estimation error. We want the estimates of ABE0 framework to represent the majority of selected instances and not greatly affected by extreme values. Therefore, ABE0 returns the median effort values of the k nearest analogies. Since ABE0 implicitly assigns equal weights to k nearest analogies, it turns out to be an U-ABE method.

In this research we will compare the results of ABE0 framework with different non-uniform weighting strategies, i.e. with different N-ABE methods. Note that since ABE0 is a framework for U-ABE methods, in the rest of the paper the two terms will be used interchangeably. N-ABE methods have been previously addressed in literature. For example inverse rank weighted mean (IRWM) was proposed by Mendes et al. [36]. IRWM method enables higher ranked analogies to have greater influence than the lower ones. Assuming that we have *3* analogies, the closest analogy (CA) gets a weight of $\frac{3}{\sum_{i=1}^{3} i}$, the second closest (SC) gets a weight of $\frac{2}{\sum_{i=1}^{3} i}$ and the last analogy (LA) gets $\frac{1}{\sum_{i=1}^{3} i}$.

4.2 Kernel Density Estimation

The kernel function is usually chosen to be unimodal and symmetric about zero [55], such as a probability distribution function. In a kernel estimation method, the center of the kernel is placed right on each data point and the influence of data points is distributed to the overall neighborhood. To reach the final density function, contributions coming from all data points are summed up. For example, note how individual Gaussian curves add up to generate the final density estimate in Figure 1 (one kernel is added for each observation along the x-axis [46]).

Kernel density estimation has been successfully used for different type of datasets. For instance Palpanas et al. use kernel density estimation to address the problem of deviation detection in environment of sensor networks [43]. Frank et al. use kernel estimation for locally weighting the attributes of Naive Bayes [13]. Furthermore John et al. use kernel estimation to tackle the normality assumption regarding continuous datasets [15].

The kernels we use in our research are: Uniform, triangular, Epanechnikov and Gaussian. We can use a generic formula for some kernels, which is given in Equation 2, where $\mathbf{1}_{(|x|<1)}$ is the indicator function. Furthermore, Equation 3 and Equation 4 explain for the calculation of other functions in Equation 2. Depending on the value of p in Equation 2, we can derive different kernels. For example for $p = 0$ we elicit the uniform kernel, for $p = 1$ we elicit Epanechnikov kernel etc.

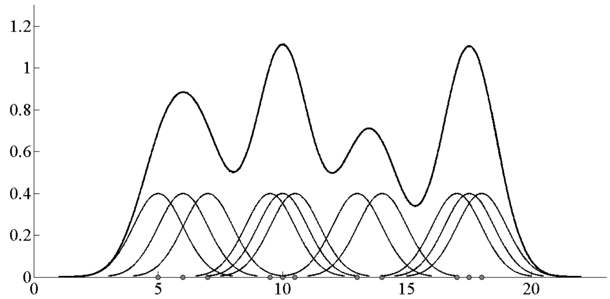


Fig. 1: We see a Gaussian kernel density estimate built on individual data points. Each point is in the center of a kernel and its effect is distributed to its neighborhood. The sum of all kernels make up the final Gaussian kernel density estimate.

$$K(x,p) = \frac{\left(1-x^2\right)^p}{2^{2p+1}B\left(p+1,p+1\right)}\mathbf{1}_{(|x|<1)} \tag{2}$$

$$B\left(p+1,p+1\right) = \frac{\Gamma\left(a\right)\Gamma\left(b\right)}{\Gamma\left(a+b\right)} \tag{3}$$

$$\Gamma\left(n\right) = (n-1)! \tag{4}$$

This paper explores the kernels of Figure 2 as well as IRWM [35,36]. IRWM is not actually proposed as a kernel method and it does not fully conform to the definition of standard kernel methods. However, due to the weighting strategy it proposes we can read it as an expert proposed kernel.

A literature review revealed that the selection of bandwidth (h) for kernels is more influential than the kernel types [10,47]. Bandwidth h is fundamentally a scaling factor

Kernel Type	**Formula**
Uniform Kernel	$K(\rho) = \frac{1}{2}\mathbf{1}_{(\|\rho\|<1)}$
Triangular Kernel	$K(\rho) = (1-\|\rho\|)\,\mathbf{1}_{(\|\rho\|<1)}$
Epanechnikov Kernel	$K(\rho) = \frac{3}{4}\left(1-\rho^2\right)\mathbf{1}_{(\|\rho\|<1)}$
Gaussian Kernel	$K(\rho) = \frac{1}{\sqrt{2\pi}}e^{\left(\frac{-1}{2}\rho^2\right)}$
IRWM Kernel	—

Fig. 2: The formulas for different kernels used in this study. In formulas $\rho = \frac{x-X_i}{h}$. Note that IRWM kernel has different characteristics and its calculation details were provided in Section 4.1.

that controls how wide probability density function will spread, i.e. appropriate choice of h is critical to avoid under and over-smoothing [46, 55]. To avoid both under and over-smoothing conditions we used various bandwidth values. One of the bandwidths we used is suggested by John et al., which is $h = 1/\sqrt{n}$ where h is the bandwidth and n is the size of dataset [15]. The other bandwidth values we used are: 2, 4, 8 and 16.

5 Methodology

5.1 Weighting Method

Assume that our dataset is divided into two sets: $A = \{x_1, ..., x_k\}$ (selected ***A***nologies) and $R = \{t_1, ..., t_{n-k}\}$ (***R***est of the dataset). We build the kernel density estimation on R and evaluate the resulting function at instances of A. Equation 5 shows the probability calculation with kernel density estimation. In Equation 5 the kernel K is built on training data $t_i \in R$ and is evaluated at k^{th} analogy (x_k) for a bandwidth of h. After scaling these probability values to 0-1 interval according to Equation 6, we use them as weights for analogies. After calculating $weight_{x_i}$ for each analogy, we update their actual effort values according to Equation 7.

$$f(x_k, h) = \frac{1}{nh} \sum_{t_i \in R} K\left(\frac{x_k - t_i}{h}\right) \tag{5}$$

$$weight_{x_i} = \frac{f(x_i, h) - max(f(x_k, h))}{max(f(x_k, h)) - min(f(x_k, h))} \tag{6}$$

$$updatedEffort_{x_i} = actualEffort_{x_i} * weight_{x_i} \tag{7}$$

5.1.1 Uniform vs. Non-Uniform Weighting

The fundamental difference between N-ABE and U-ABE methods is that in U-ABE analogies are given uniform weights and their actual effort values are used in an *as is* manner, whereas in N-ABE analogies are assigned different weights and their actual effort values are multiplied by these weight values. As for U-ABE, we defined a base method that we call ABE0 and for N-ABE we use 5 different kernel methods.

One point that needs further clarification is the use of uniform kernel as a N-ABE method. Figure 3 succinctly illustrates the difference between uniform kernel being a N-ABE method and ABE0 being a U-ABE method. ABE0 assumes equal importance of *all* instances and assigns equal probabilities. A uniform kernel would assign equal non-zero probabilities to *only a certain portion* of the instances, whereas the rest of the instances would be assigned a weight of zero (i.e. they would be ignored).

5.2 Data

In our research, we have used 19 datasets, most of which are heavily used in software effort estimation research: Nasa93, the original Cocomo81 [6], Desharnais [11] and so on. Note that 4 projects in Desharnais dataset has missing entries, we used imputation [1] to handle them. The details regarding these datasets can be found in Figure 4.

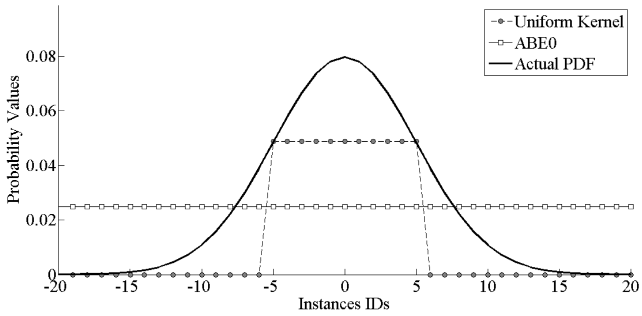


Fig. 3: In the case of ABE0 all instances are given equal probability values, hence equal weights. However, uniform kernel prefers some instances over the others: Only a certain portion of the instances are given equal non-zero weights.

Dataset	**Features**	**Size**	**Description**	**Units**
Cocomo81	17	63	NASA projects	months
Cocomo81e	17	28	Cocomo81 embedded projects	months
Cocomo81o	17	24	Cocomo81 organic projects	months
Cocomo81s	17	11	Cocomo81 semi-detached projects	months
Nasa93	17	93	NASA projects	months
Nasa93_center_1	17	12	Nasa93 projects from center 1	months
Nasa93_center_2	17	37	Nasa93 projects from center 2	months
Nasa93_center_5	17	40	Nasa93 projects from center 5	months
Desharnais	12	81	Canadian software projects	hours
DesharnaisL1	11	46	Desharnais projects using Language1	hours
DesharnaisL2	11	25	Desharnais projects using Language2	hours
DesharnaisL3	11	10	Desharnais projects using Language3	hours
SDR	22	24	Turkish software projects	months
Albrecht	7	24	Projects from IBM	months
Finnish	8	38	Finland projects	hours
Kemerer	7	15	Large business applications	months
Maxwell	27	62	Projects from commercial banks in Finland	hours
Miyazaki94	8	48	Japanese COBOL projects	months
Telecom	3	18	Maintenance projects for telecom companies	months
	Total	699		

Fig. 4: We used 699 projects coming from 19 datasets. Datasets have different characteristics in terms of the number of attributes as well as the measures of these attributes.

5.3 Experiments

Our experimental settings aim at comparing the performance of standard U-ABE (ABE0) to that of N-ABE. To separate train and test sets we use leave-one-out method, which entails selecting 1 instance out of a dataset of size n as the test set and using the remaining $n-1$ instances as the training set. For each test instance, we run ABE0 and N-ABE separately and store their estimates. As the analogy number is reported to play a critical role in estimation accuracy [18], both for U-ABE and N-ABE methods, we tried different k values. Furthermore, to hinder any particular bias that would come from the settings of a single experiment, we repeated the afore mentioned procedure 20 times.

We use 2 forms of ABE methods (uniform and non-uniform weighting) induced on 19 datasets for 5 different k values. The k values we used in our research are: $k \in \{3, 5, 7, 9, best\}$. $best$ is a pseudo-best k value that is selected for each individual test instance through a process, in which we randomly pick up 10 instances from the training set and select the lowest error yielding k value as the $best$. Note that $k > 1$ for $\forall k$, because for $k = 1$ U-ABE and N-ABE would be equivalent. In addition, we use 5 different kernels (Uniform, Triangular, Epanechnikov, Gaussian and IRWM) with 5 bandwidth values in N-ABE experiments. To further explore field of software effort estimation, we investigate a total of 2090 different settings in this research:

- U-ABE Experiments: 95 settings
  - *19 datasets * 5 k values = 95*
- N-ABE Experiments: 1995 settings
  - Standard Kernels: *19 datasets * 5 k values * 4 kernels * 5 bandwidths = 1900*
  - IRWM: *19 datasets * 5 k values = 95*

5.4 Performance Criteria

Performance measures comment on the success of a prediction. For example, mean absolute residual (MAR) is the mean of absolute residuals (the difference between the predicted and the actual):

$$MAR = mean(AR_i = |actual_i - predicted_i|) \tag{8}$$

Magnitude of relative error (MRE) is one of the most commonly used performance criterion for assessing the performance of competing software effort estimation methods [8, 12, 40].

$$MRE = \frac{|actual_i - predicted_i|}{actual_i} \tag{9}$$

Median MRE (MdMRE) has emerged as the de facto standard evaluation criterion for cost estimation models [52]. Median also gives information about central tendency and is less sensitive to extreme MRE values. MdMRE formula is given in Equation 10, where n is the test set size.

$$MdMRE = median(MRE_1, MRE_2, ..., MRE_n) \tag{10}$$

Another alternative performance measure is PRED(25), which is reported to be one of the most commonly used accuracy statistics [26]. It is defined as the percentage of predictions falling within 25% of the actual values [34]:

$$PRED(25) = \frac{100}{N} \sum_{i=1}^{N} \begin{cases} 1 \text{ if } MRE_i \leq \frac{25}{100} \\ 0 \text{ otherwise} \end{cases} \tag{11}$$

For example, PRED(25)=50% implies that half of the estimates are failing within 25% of the actual values [50].

If performance measures are used as a stand-alone evaluation criteria (i.e. not combined with appropriate statistical tests), results may lead to biased or even false conclusions [12]. Therefore, we use the so called $win, tie, loss$ statistics, whose pseudocode

```
if Mann-Whitney(P_i, P_j, 95) says they are the same then
  tie_i = tie_i + 1;
  tie_j = tie_j + 1;
else
  if better( median(P_i), median(P_j)) then
    win_i = win_i + 1
    loss_j = loss_j + 1
  else
    win_j = win_j + 1
    loss_i = loss_i + 1
  end if
end if
```

Fig. 5: Comparing algorithms (i,j) on performance (P_i,P_j). The "better" predicate changes according to P. For error measures like MRE, "better" means lower medians. However, for PRED(25), "better" means higher medians.

is given in Figure 5. In Figure 5, we first check if two distributions i, j are statistically different (Mann-Whitney rank-sum test, 95% confidence); otherwise we increment tie_i and tie_j. If the distributions are statistically different, we update win_i, win_j and $loss_i, loss_j$ after comparing the performance measures so as to see which one is better.

6 Results

To see the effect of kernel weighting, we studied 19 datasets and 3 different performance measures. For each performance measure we tried 4 different kernels subject to 5 different bandwidths, plus the IRWM kernel (which does not have a bandwidth concept) and reported associated $win, tie, loss$ statistics.

Figure 6 shows a sample of our results. It reports the $win, tie, loss$ statistics of Desharnais dataset for ABE0 and N-ABE through Gaussian kernel. For each dataset we have 4 such tables (one for each kernel), so for all datasets there are *19 Datasets* $\times$ *4 tables = 76 tables*. For the IRWM kernel there will be another *19 datasets* $\times$ *1 kernel = 19 tables*. Hence, in total, our results comprise *76 + 19 = 95 tables* to report. All these tables are available on line at http://goo.gl/qpQiD. However, for space reasons, we summarize those tables as follows.

In Figure 6 we see that each row reports $win, tie, loss$ statistics of ABE0 methods (*k=3,5,7,9,best*) as well as N-ABE methods (*k=[3,5,7,9,best]+kern* where *kern* stands for kernel weighting) subject to a particular performance measure. Similarly, each column shows the $win, tie, loss$ statistics associated with a particular bandwidth value. As can be seen in Figure 6, for Desharnais dataset ABE0 methods always have higher *win* values and always have a *loss* value of 0, meaning that they never lose against N-ABE methods. That is, by summarizing each row/column intersection we can see that the performance of ABE0 is never improved by N-ABE.

Figure 7 and Figure 8 repeat that summarization process for all 19 datasets and all kernel/bandwidth combination:

- Each row of these summary figures shows the comparison of ABE0 performance to that of N-ABE subject to 3 different performance measures.
- Every kernel/bandwidth intersection in Figure 7 and Figure 8 has 3 symbols corresponding to MdMRE, MAR and Pred(25) comparisons from left to right.
- Each of these 3 symbols can have 3 values: $-, +, o$.

		h=1/sqrt(size)			h=2			h=4			h=8			h=16		
		WIN	TIE	LOSS	WIN	TIE	LOSS	WIN	TIE	LOSS	WIN	TIE	LOSS	WIN	TIE	LOSS
MdMRE	k=3	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=5	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=7	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=9	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=best	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=3+kern	80	0	100	80	0	100	80	0	100	80	0	100	80	0	100
	k=5+kern	0	60	120	60	0	120	60	0	120	60	0	120	60	0	120
	k=7+kern	0	60	120	20	20	140	20	20	140	20	20	140	20	20	140
	k=9+kern	0	60	120	0	0	180	0	0	180	0	0	180	0	0	180
	k=best+kern	0	60	120	20	20	140	20	20	140	20	20	140	20	20	140
MAR	k=3	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=5	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=7	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=9	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=best	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=3+kern	0	80	100	60	20	100	60	20	100	60	20	100	60	20	100
	k=5+kern	0	80	100	0	80	100	0	80	100	0	80	100	0	80	100
	k=7+kern	0	80	100	0	60	120	0	60	120	0	60	120	0	60	120
	k=9+kern	0	80	100	0	60	120	0	60	120	0	60	120	0	60	120
	k=best+kern	0	80	100	0	60	120	0	60	120	0	60	120	0	60	120
Pred(25)	k=3	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=5	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=7	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=9	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=best	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=3+kern	60	0	120	80	0	100	80	0	100	80	0	100	80	0	100
	k=5+kern	20	60	100	60	0	120	60	0	120	60	0	120	60	0	120
	k=7+kern	0	60	120	0	20	160	0	20	160	0	20	160	0	20	160
	k=9+kern	0	60	120	20	0	160	20	0	160	20	0	160	20	0	160
	k=best+kern	0	60	120	20	20	140	20	20	140	20	20	140	20	20	140

Fig. 6: Desharnais dataset $win, tie, loss$ statistics for ABE0 and N-ABE through Gaussian kernel. For each dataset we have 4 of these tables (one for each kernel). In total it amounts to *19 Datasets × 4 tables = 76 tables*. In addition we have another *19 datasets × 1 kernel = 19 tables* from IRWM kernel. It is infeasible to include all the tables in this paper, therefore an executive summary of $76 + 19 = 95$ tables is provided in Figure 7. Furthermore, we provide all 95 tables in excel format at http://goo.gl/qpQiD.

Dataset	**Kernel**	**h=1/sqrt(size)**	**h = 2**	**h = 4**	**h = 8**	**h = 16**
Coc81	**Uniform**	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Coc81e	**Uniform**	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Coc81o	**Uniform**	$-o-$	$-oo$	$-oo$	$-oo$	$-oo$
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	$---$	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	$-o-$	*ooo*	*ooo*	*ooo*	*ooo*
Coc81s	**Uniform**	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Ns93	**Uniform**	$-o-$	$-o-$	$-o-$	$-o-$	$-o-$
	Triangular	$-o-$	$-o-$	$-o-$	$-o-$	$-o-$
	Epanechnikov	$-o-$	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	$-o-$	*ooo*	*ooo*	*ooo*	*ooo*
Ns93c1	**Uniform**	$---$	$---$	$---$	$---$	$---$
	Triangular	$---$	$-o-$	$-o-$	$-o-$	$-o-$
	Epanechnikov	$---$	$-o-$	$-o-$	$-o-$	$-o-$
	Gaussian	$-o-$	$-o-$	$-o-$	$-o-$	$-o-$
Ns93c2	**Uniform**	*ooo*	$-o-$	$-o-$	$-o-$	$-o-$
	Triangular	$-o-$	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	$-o-$	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Ns93c5	**Uniform**	$---$	$-o-$	$-o-$	$-o-$	$-o-$
	Triangular	$---$	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	$---$	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	$---$	*ooo*	*ooo*	*ooo*	*ooo*

Fig. 7: Nine data sets comparing ABE0 to N-ABE. For every row in each cell, there are three symbols indicating the effect of N-ABE w.r.t. 3 different error measures. From left to right, the first symbol stands for N-ABE effect w.r.t. MdMRE, the second symbol w.r.t. MAR and the third one w.r.t. Pred(25). A "+" indicates that for majority of k values (at least 3 out of 5 k values), N-ABE improved ABE0 in terms of $win - loss$ values. "$-$" indicates that N-ABE decreased the performance of ABE0 in the majority case. If the former conditions do not satisfy, then a "o" symbol is assigned. Note that dataset order here is the same as Figure 4, yet the dataset names are abbreviated to 3 to 5 letters due to space constraints.

- *"$-$"* means that N-ABE decreased the accuracy of ABE0;
- *"o"* means ABE0 and N-ABE are statistically same;
- *"+"* shows that ABE0 accuracy was improved through kernel weighting (i.e. N-ABE has a better performance than ABE0).

We assign the symbols "+" or "$-$" if the performance associated with the majority of the k-values (at least 3 out of 5) are improved or degraded by N-ABE (in terms of $win - loss$). If there is no change, we assign a "o" symbol to that setting.

Observe that in all these summaries:

- There is only one dataset (SDR in Figure 8) where N-ABE provides a performance improvement in certain cases. Even for that dataset there are 15 "+" symbols and 21 "$-$" symbols, meaning that most of the time N-ABE is still destructive.
- In 18 other datasets, which is $\frac{18}{19} = 95\%$ of all the datasets, there is not a single case where N-ABE improves the performance of ABE0.

Dataset	Kernel	h=1/sqrt(size)	h = 2	h = 4	h = 8	h = 16
Des	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	ooo	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	− − −	− − −	− − −	− − −
	Gaussian	− − −	− − −	− − −	− − −	− − −
DesL1	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	ooo	−o−	−o−	−o−	−o−
	Epanechnikov	− − −	−o−	−o−	−o−	−o−
	Gaussian	− − −	−o−	−o−	−o−	−o−
DesL2	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	ooo	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	−o−	−o−	−o−	−o−
	Gaussian	− − −	−o−	−o−	−o−	−o−
DesL3	**Uniform**	−o−	−o−	−o−	−o−	−o−
	Triangular	−o−	ooo	ooo	ooo	ooo
	Epanechnikov	−o−	ooo	ooo	ooo	ooo
	Gaussian	−o−	ooo	ooo	ooo	ooo
SDR	**Uniform**	−o−	−o−	−o−	−o−	−o−
	Triangular	+ + −	+o−	+o−	+o−	+o−
	Epanechnikov	− + −	+ + −	+ + −	+ + −	+ + −
	Gaussian					
Albr	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	− − −	−o−	−o−	−o−	−o−
	Epanechnikov	− − −	−o−	−o−	−o−	−o−
	Gaussian	− − −	−o−	−o−	−o−	−o−
Finn	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	ooo	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	−o−	−o−	−o−	−o−
	Gaussian	− − −	−o−	−o−	−o−	−o−
Kem	**Uniform**	−o−	−o−	−o−	−o−	−o−
	Triangular	− − −	ooo	ooo	ooo	ooo
	Epanechnikov	− − −	ooo	ooo	ooo	ooo
	Gaussian	− − −	ooo	ooo	ooo	ooo
Maxw	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	− − −	ooo	ooo	ooo	ooo
	Epanechnikov	− − −	ooo	ooo	ooo	ooo
	Gaussian	− − −	ooo	ooo	ooo	ooo
Miy94	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	− − −	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	−o−	−o−	−o−	−o−
	Gaussian	− − −	−o−	−o−	−o−	−o−
Tel	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	−o−	−o−	−o−	−o−	−o−
	Epanechnikov	− − −	−o−	−o−	−o−	−o−
	Gaussian	− − −	−o−	−o−	−o−	−o−

Fig. 8: Ten more data sets comparing ABE0 to N-ABE. Same format as Figure 7.

Note that these summary tables contain results from different performance criteria (MdMRE, MAR, Pred(25)) as well as kernels and bandwidths. Therefore, our conclusion from Figure 7 and Figure 8 is that that *"non-uniform weighting through standard kernel methods does not improve the performance of ABE"* holds in the majority case across different datasets and error measures.

Another summary table is given in Figure 9. Figure 9 is very similar to Figure 7 in the sense that it summarizes the performance of N-ABE over 19 datasets w.r.t. three different performance measures. The difference is that Figure 7 summarizes the results of standard kernel methods, whereas in Figure 9 we see the N-ABE performance under an *expert-based* kernel, i.e. IRWM. Although there are important differences between standard and expert-based kernels (IRWM has no bandwidth parameter), the results seen in Figure 9 is quite similar to those of Figure 7. As can be seen in Figure 9, there is not a single case where N-ABE (under IRWM kernel) improves the performance of

Dataset	Improvement		
	MdMRE	MAR	Pred(25)
Cocomo81	$-$	o	$-$
Cocomo8e	o	o	o
Cocomo8o	$-$	$-$	$-$
Cocomo8s	o	o	o
Nasa93	$-$	$-$	$-$
Nasa93_center_1	$-$	$-$	$-$
Nasa93_center_2	$-$	o	$-$
Nasa93_center_5	$-$	$-$	$-$
Desharnais	$-$	$-$	$-$
DesharnaisL1	$-$	$-$	$-$
DesharnaisL2	$-$	$-$	$-$
DesharnaisL3	$-$	o	$-$
SDR	$-$	o	$-$
Albrecht	$-$	$-$	$-$
Finnish	$-$	$-$	$-$
Kemerer	$-$	$-$	$-$
Maxwell	$-$	$-$	$-$
Miyazaki94	$-$	$-$	$-$
Telecom	$-$	$-$	$-$

Fig. 9: The comparison of ABE0 to N-ABE under IRWM kernel. Similar to Figure 7 three symbols indicate the effect of N-ABE w.r.t. 3 different error measures and "+" indicates that for majority of k values N-ABE improved ABE0 in terms of $win - loss$ values. A "$-$" symbol indicates a decrease and a "o" symbol indicates neither decrease nor increase. Notice that subject to IRWM kernel, N-ABE fails to improve ABE0 w.r.t. 3 different performance measures.

ABE0. Furthermore, the amount of "$-$" symbols is much more than "o", meaning that N-ABE decreases the performance of ABE0 most of the time.

7 Discussion

Our pre-experimental intuition was that non-uniform weighting could end up performing no better and perhaps even worse than simple uniform weighting for effort estimation datasets. A simple example for such a negative intuition is provided in Figure 10.

Analogies	P_1	P_2	P_3
Effort Values	10.0	20.0	60.0
Probabilities	0.1	0.3	0.5
Weights	0.0	0.5	1.0
Uniform Weighting Estimate	20.0		
Non-Uniform Weighting Estimate	35.0		

Fig. 10: An intuitive example. In a 3 analogy case, there is a 75% change for a hypothetical test project between uniform and non-uniform weighting.

Assume that for a test project in Figure 10, 3 analogies are chosen (P_1, P_2, P_3) with effort values of 10.0, 20.0 and 60.0. Also assume that a kernel assigned the probabilities of 0.1, 0.3 and 0.5 to these analogies respectively. When these probability values are normalized, the weights assigned to P_1, P_2, P_3 become 0.0, 0.5 and 1.0. The estimation (i.e. median of analogies) for uniform weighting case would be 20.0, whereas the estimation for non-uniform weighting case would be 35.0. A shift from an estimate of 20.0 to 35.0 is a dramatic change of 75% rather than a slight correction. Therefore, our intuition was that non-uniform weighting in software effort estimation could be disruptive rather than constructive.

Then the most likely question to be raised is “Why do other fields [13,15,43] benefit from weighting, whereas effort estimation does not?”. Our belief is that the answer is partially hidden behind the low sample sizes of effort datasets. Scarcity of the samples means that the weighting observes a signal being broadcast from a very small number of points in the neighborhood. In Figure 11 we simulate 50, 100 and 1000 samples coming from two Gaussian probability distribution functions (PDFs): $N(20, 5)$ and $N(35, 5)$. Then we use kernel density estimation technique with a Gaussian kernel to estimate the density at discrete values of x in [0-55] interval with a step-size of 1.

In Figure 11, closest estimates require:

- Optimum bandwidth (here $h = 1$). Too small bandwidth ($h = 0.001$) assigns most probability values (hence weights) to zero, whereas too big of a bandwidth ($h = 10$) results in over-smoothing.
- Considerable sample size. Note how optimum fit is achieved for a sample size of 1000.

In case of signal processing the sample sizes are closer to Figure 11(c) and as we see from the simulation example, kernel estimates can successfully model such densely populated datasets. However, software effort datasets used in our research are similar to Figure 11(a) and Figure 11(b). When we observe behavior of kernel estimates for low sample sizes in those figures, it is somewhat expected to see lower performance values in sparsely populated data sets like software effort data sets.

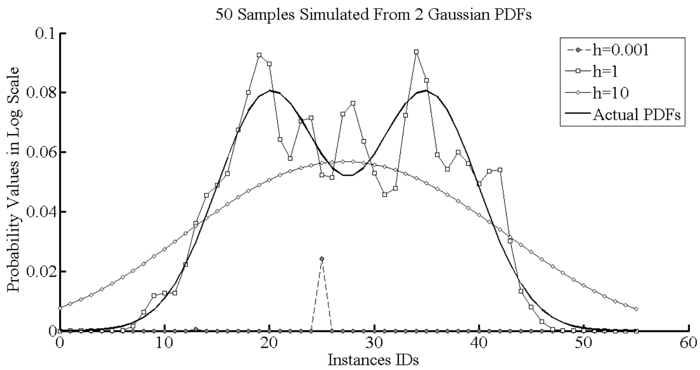
8 Threats to Validity

We will address the threats to validity of this research under 3 categories: Internal validity, external validity and construct validity.

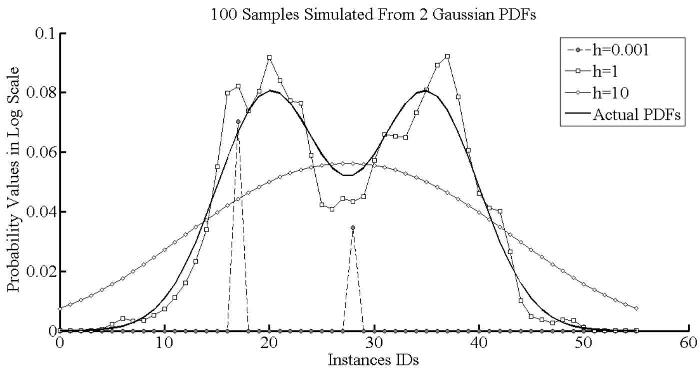
Internal validity asks to what extent the cause-effect relationship between dependent and independent variables holds [1]. We use leave-one-out method for all treatments to address internal validity issues. Leave-one-out selection enables us to separate the training and test sets completely in each experiment, thereby making the test sets completely new situations for the training sets.

External validity questions the ability to generalize the results [39]. To observe the generalizability of our results, we perform extensive experiments on 19 datasets. The datasets are widely used in software effort estimation community and have very different characteristics in terms of various criteria such as size, number of features, types of features and measurement method. However, to have full confidence in our claims, our study needs to be replicated by future studies.

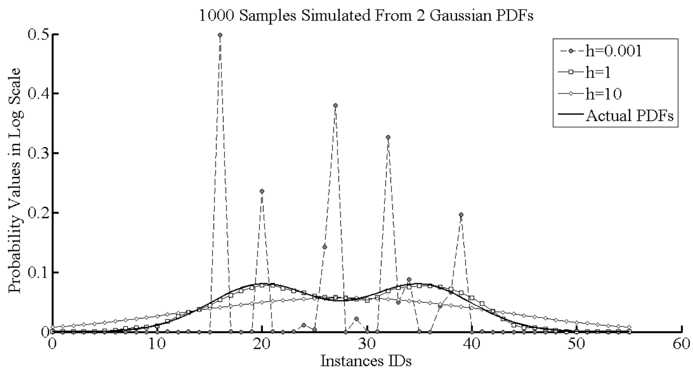
Construct validity (i.e. face validity) makes sure that we in fact measure what we intend to measure [45]. Kitchenham et al. notes that different performance measures evaluate different aspects of prediction accuracy [26]. So as to assess N-ABE and ABE0 comparison from different standpoints, we made use of MAR, MdMRE and Pred(25) in our study. However, as Kitchenham et al. points out in [25], it is wrong to solely use the performance measures without a statistical test. Therefore, we also use win-tie-loss measures, where we make use of Mann-Whitney U test at a significance level of 95%.



(a) 50 Sample Points: Note the bad fit due to low sample size.



(b) 100 Sample Points: Note the better fit due to increased sample size.



(c) 1000 Sample Points: Note the optimum fit due to high sample size.

Fig. 11: The effect of sample size and bandwidth on kernel density estimation. The choice of optimum bandwidth (h value) is important. However, even with the optimum bandwidth, one still needs enough number of samples for successful estimation. Sample size of 50 appears to be too small and when we increase it to 100, we get a better fit. Yet, for a very close fit, we need to go up to 1000 sample points.

9 Conclusions

In this research we tried kernel density estimation as a non-uniform weighting strategy for ABE. We conducted experiments with various kernels as well as bandwidths. For the datasets and performance measures used in our research there was hardly any cases where N-ABE methods outperformed ABE0. We hesitate to discourage further research with different experimental settings or with different datasets. However, if similar use of kernels is to be adopted, we do not recommend the use of kernel methods as a weighting strategy in ABE.

Unlike studies in different domains that use kernel methods and report improved accuracy values [15, 43], we do not observe such an effect on software effort datasets. The reason for different results may lie in different dataset characteristics. For instance the datasets used in other domains are much more densely populated than software effort datasets.

9.1 Answers To Research Questions

RQ1. Is there evidence that non-uniform weighting improves the performance of ABE? The results of our experiments do not show such an evidence. On the contrary, for all settings ABE0 yields much better results than N-ABE methods.

RQ2. What is the effect of different kernels for non-uniform weighting in ABE? There are only slight variations in performance when different kernels are used. However, these variations do not follow a definite pattern and they are far from being considerable.

RQ3. What is the effect of different bandwidths? Change of bandwidths shows a random and insignificant effect, which is very similar to that of kernel change effect. Therefore, we cannot say that applying different bandwidths has a certain effect on N-ABE performance.

RQ4. How do the characteristics of software effort datasets influence the performance of kernel weighting in N-ABE? Effort datasets are much smaller than most of the datasets in different domains. The dependent variable (effort value of a completed project) is highly variable. Furthermore, the attribute values are very open to personal judgment and error. All these factors suggest that non-parametric methods may be failing due to inherent characteristics of software effort data.

10 Future Work

We can identify 3 main domains in which this study may be extended:

1. *Dataset:* In this research we used 19 software effort datasets. However, this study may be replicated on other software effort datasets as well.
2. *Weighting Strategy:* Another future direction can be experimentation on different non-uniform weighting strategies that are preferably based on different assumptions.
3. *Customization:* Experimentation in this paper is based on publicly available datasets. Our knowledge about the context and assumptions of these datasets are limited to their documentation. One further direction to this research could be customizing

this work for a particular company and derive the actual problems associated with non-uniform weighting schemes.

The experiments shown in this research took three months to research, design, execute, then write up. It turns out that we could have spent the time more productively on other issues. Our pre-experimental intuition that "non-uniform weighting in the data sparse domain of software effort estimation may not provide an improvement in estimation accuracy" turned out to be correct. We want to remind researchers, who want to follow afore mentioned future directions that those future directions may as well end up in negative results.

References

1. E. Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.
2. L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical Softw. Engg.*, 5(1):35–68, 2000.
3. M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl. Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. *IEEE Trans. Softw. Eng.*, 32:83–92, 2006.
4. D. Baker. A hybrid approach to expert and model-based effort estimation. Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2007.
5. B. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches: A survey. *Annals of Software Engineering*, 10:177–205, 2000.
6. B. W. Boehm. *Software Engineering Economics.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
7. L. Briand, K. El Emam, and F. Bomarius. Cobra: a hybrid method for software cost estimation, benchmarking, and risk assessment. In *Proceedings of the International Conference on Software Engineering*, pages 390 –399, April 1998.
8. L. C. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. D. Maxwell. An assessment and comparison of common software cost estimation modeling techniques. pages 313–322, 1999.
9. H. I. Browman. Negative results. *Mar. Ecol. Prog. Ser.*, 191:301–309, 1999.
10. N. A. C. Cressie. *Statistics for Spatial Data (Wiley Series in Probability and Statistics).* Wiley-Interscience, 1993.
11. J. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. Master's thesis, Univ. of Montreal, 1989.
12. T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Trans. Softw. Eng.*, vol:29no11pp985–995, 2003.
13. E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.
14. R. Jeffery, M. Ruhe, and I. Wieczorek. Using public domain metrics to estimate software development effort. In *METRICS '01: Proceedings of the 7th International Symposium on Software Metrics*, page 16, Washington, DC, USA, 2001. IEEE Computer Society.
15. G. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.
16. M. Jorgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70:37–60, February 2004.
17. M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Trans. Softw. Eng.*, 33(1):33–53, 2007.
18. G. Kadoda, M. Cartwright, and M. Shepperd. On configuring a case-based reasoning software project prediction system. *UK CBR Workshop, Cambridge, UK*, pages 1–10, 2000.
19. C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, May 1987.

20. J. Keung. Empirical evaluation of analogy-x for software cost estimation. In *ESEM '08: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 294–296, New York, NY, USA, 2008. ACM.
21. J. Keung and B. Kitchenham. Experiments with analogy-x for software cost estimation. In *ASWEC '08: Proceedings of the 19th Australian Conference on Software Engineering*, pages 229–238, Washington, DC, USA, 2008. IEEE Computer Society.
22. J. Keung, E. Kocaguneli, and T. Menzies. A ranking stability indicator for selecting the best estimator in software cost estimation. *Automated Software Engineering (under review)*, 2011. `http://menzies.us/pdf/11draftranking.pdf`.
23. J. W. Keung, B. A. Kitchenham, and D. R. Jeffery. Analogy-x: Providing statistical inference to analogy-based software cost estimation. *IEEE Trans. Softw. Eng.*, 34(4):471–484, 2008.
24. C. Kirsopp, M. Shepperd, and R. House. Case and feature subset selection in case-based software project effort prediction. *Research and development in intelligent systems XIX: proceedings of ES2002, the twenty-second SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, page 61, 2003.
25. B. Kitchenham and E. Mendes. Why comparative effort prediction studies may be invalid. In *PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, pages 1–5, New York, NY, USA, 2009. ACM.
26. B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd. What accuracy statistics really measure [software estimation]. *Software, IEE Proceedings -*, 148(3):81 –85, June 2001.
27. M. Kläs, A. Trendowicz, A. Wickenkamp, J. Münch, N. Kikuchi, and Y. Ishigai. The use of simulation techniques for hybrid software cost estimation and risk analysis. *Advances in Computers*, 74:115–174, 2008.
28. E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung. Exploiting the essential assumptions of analogy-based effort estimation. *IEEE Trans. Softw. Eng.*, 2011 (to appear). `http://menzies.us/pdf/10teak-v1.pdf`.
29. J. Li and G. Ruhe. A comparative study of attribute weighting heuristics for effort estimation by analogy. *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, page 74, 2006.
30. J. Li and G. Ruhe. Analysis of attribute weighting heuristics for analogy-based software effort estimation method aqua+. *Empirical Software Engineering*, 13:63–96, 2008.
31. J. Li, G. Ruhe, A. Al-emran, and M. M. Richter. A flexible method for software effort estimation by analogy. *Empirical Software Engineering*, pages 65–106, 2007.
32. Y. Li, M. Xie, and T. Goh. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82:241–252, 2009.
33. E. Mendes and N. Mosley. Further investigation into the use of cbr and stepwise regression to predict development effort for web hypermedia applications. *Empirical Software Engineering, International Symposium on*, 0:79, 2002.
34. E. Mendes and N. Mosley. Bayesian network models for web effort prediction: A comparative study. *IEEE Trans. Softw. Eng.*, 34:723–737, 2008.
35. E. Mendes, N. Mosley, and I. Watson. A comparison of case-based reasoning approaches. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 272–280, New York, NY, USA, 2002. ACM.
36. E. Mendes, I. D. Watson, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.
37. T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Trans. Softw. Eng.*, 32:883–895, 2006.
38. T. Menzies, O. Jalali, J. Hihn, D. Baker, and K. Lum. Stable rankings for different effort models. *Automated Software Engg.*, 17:409–437, December 2010.
39. D. Milic and C. Wohlin. Distribution Patterns of Effort Estimations. In *Euromicro*, 2004.
40. K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove. A survey on software estimation in the norwegian industry. *IEEE International Symposium on Software Metrics*, pages 208–219, 2004.
41. I. Myrtveit and E. Stensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Trans. Softw. Eng.*, 25:510–525, 1999.
42. I. Myrtveit, E. Stensrud, and M. Shepperd. Reliability and validity in comparative studies of software prediction models. *IEEE Trans. Softw. Eng.*, vol:31no5pp380–391, May 2005.

43. T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Rec*, 32:2003, 2003.
44. P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger. A probabilistic model for predicting software development effort. *IEEE Trans. Softw. Eng.*, 31:615–624, 2005.
45. C. Robson. Real world research: a resource for social scientists and practitioner-researchers. *Blackwell Publisher Ltd*, 2002.
46. S. Scheid. Introduction to kernel smoothing. Talk, January 2004.
47. D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.
48. M. Shepperd. Software project economics: a roadmap. In *FOSE '07: 2007 Future of Software Engineering*, pages 304–315, 2007.
49. M. Shepperd and G. Kadoda. Comparing software prediction models using simulation. *IEEE Trans. Softw. Eng.*, pages 1014–1022, 2001.
50. M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Trans. Softw. Eng.*, 23(11):736–743, 1997.
51. M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In *International Conference on Software Engineering*, pages 170–178, 1996.
52. E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. An empirical validation of the relationship between the magnitude of relative error and project size. In *METRICS '02: Proceedings of the 8th International Symposium on Software Metrics*, page 3, Washington, DC, USA, 2002. IEEE Computer Society.
53. A. Trendowicz, J. Heidrich, J. Münch, Y. Ishigai, K. Yokoyama, and N. Kikuchi. Development of a hybrid cost estimation model in an iterative manner. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 331–340, New York, NY, USA, 2006. ACM.
54. F. Walkerden and R. Jeffery. An empirical study of analogy-based software effort estimation. *Empirical Software Engineering*, 4(2):135–158, 1999.
55. M. P. Wand and M. C. Jones. *Kernel Smoothing (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, December 1994.

Replies to reviewer comments

Thank you for your detailed reviews on the previous version. They were most helpful. Replies to comments are below.

Editor

The empirical study produces a negative result - which is generally quite welcome. However, the reviewers (and I) question the generality of your conclusions! Does your result mean that "Kernel density estimation" must not be investigated any further, or...

We completely agree with your concerns- the empirical basis of the prior draft was too restrictive for us. In this version, we repeated our study on 19 data sets (not the 3 we saw before) and the same result holds.

Also, we were somewhat too adamant in the previous draft. We made our suggestion/conclusion more clear in this version. The last sentence of Abstract reads:

> "Hence, -provided that similar experimental settings are adopted- we discourage the use of kernel methods as a weighting strategy in ABE. "

Also the last sentences of the 1^{st} paragraph in Section 9 states:

> "We hesitate to discourage further research with different experimental settings or with different datasets. However, if similar use of kernels is to be adopted, we do not recommend the use of kernel methods as a weighting strategy in ABE."

This makes it clear that we are only talking about ABE for effort estimation and not (as might be read in the previous draft), kernel methods for all SE applications.

If you think that you can accommodate these concerns (especially the concern of Reviewer 2), please submit a major revision".

We highly value your invitation for a re-submit after a major revision. Therefore, we paid particular attention to address every comment of reviewers. After following through the invaluable comments of reviewers, this study has become much more sound and to the point. Furthermore, we improved the study with further experiments: Previous version used 3 datasets evaluated with 2 performance measures, whereas this version makes use of 19 datasets evaluated with 3 error measures; previous version used 330 ABE variants, whereas this version reports 2090 variants. We believe that it is now ready for a re-review. Below is a list of our replies to every individual comment.

Reviewer 1:

The article describes an analysis of a specific type of analogy-based estimation, i.e., kernel-density estimation for non-uniform weights of analogies. The article is very well written and structured. It is easy to read.

Thank you very much, we are very pleased with your comments.

Although the article focuses on an important topic, I have several concerns:

We fully agree with all your concerns. We went through all your comments one by one and restructured our work accordingly. Following is a list of our replies.

The motivation why kernel-density estimation was chosen as object of study is quite weak and not convincing to me as motivation for performing a comprehensive study. The more obvious reason that the technique could be specifically suited for software engineering projects and data is not addressed.

The previous paper was, as you say, very weak on motivation. To repair that, we have added new motivation section (Section 2) that explains why we performed this study.

The statement "ABE methods are never enhanced by a particular kernel method" seems to be based on some kind of proof. However, it could also stem from your observations. I recommend to describe in more detail what the (potentially empirical basis) for this statement is.

You are right, we should have made that statement more clear. Our claim depends on empirical evaluation of 2090 ABE variants (it was 330 in the previous draft) according to various performance criteria such as MdMRE, Pred(25) and win-tie-loss values. Furthermore, in this version we included MAR criterion in accordance with the feedback we received from another reviewer. We explain this claim very early, now the last two sentence of Abstract reads as:

> After an extensive experimentation of 19 datasets, 3 evaluation criteria, 5 kernels, 5 bandwidth values and a total of 2090 ABE variants, we found that: 1) non-uniform weighting through kernel methods cannot outperform uniform weighting ABE and 2) kernel type and bandwidth parameters do not produce a definite effect on estimation performance. Hence, -provided that similar experimental settings are adopted- we discourage the use of kernel methods as a weighting strategy in ABE.

*Section 2.1 I recommend to also integrate hybrid methods (i.e., methods based on data *and* expert judgment such as CoBRA. I also recommend to include some studies that show that the accuracy can be significantly improved by maintaining the estimation model over time. It would be nice for the reader if such kind of concept is also part of your work.*

Thank you very much for reminding us an important background work. We happily included CoBRA like integrated approaches and their iterative applications. Now the last sentences of the first paragraph in Section 4.1 is:

> There are also successful applications where expert and model based techniques are integrated to complement one another [7,21]. In particular when such estimation practices are employed iteratively over time, the estimation accuracy can significantly be improved. For example in [40], an integrated approach called CoBRA was applied in an iterative manner and estimation accuracy was improved from 120% error down to 14%.

The so-called "experiments" are mainly data analyses of existing repositories. This seems to be reasonable to a certain extend. However, it is quite a weak evaluation due to a lack of knowledge about context and assumptions. I recommend for future work to do a customization for a specific company and to derive real problems afterwards.

We agree that analysis in a company setting where we would tackle with a real problem would be a very interesting future direction. Therefore, we included it as the third element of our Future Work Section. Now the third future direction in the paper is:

> 3. Customization: Experimentation in this paper is based on publicly available datasets. Our knowledge about the context and assumptions of these datasets are limited to their documentation. One further direction to this research could be customizing this work for a particular company and derive the actual problems associated with non-uniform weighting schemes.

Overall, the scope (technique, purposes, domain) is very limited and probably only of minor interest for most of the EMSE journal readers.

We agree with the reviewer that this paper will be of interest only to those working in the effort estimation area.

However, those researchers do publish in EMSE. Jorgensen and Shepperd [17] report that of the 76 journals that published effort estimation results, this journal is ranked in the top four (they list a dozen EMSE publications related to effort estimation). Also, this query "`http://www.springerlink.com/content/1382-3256/?k=cost+estimation&o=10`" yields dozens of papers relating to cost/effort estimation in this journal.

Reviewer 2:

First of all I would like to applaud the authors' candor about the negative results rather than trying to dress them up in some way.

Thank you very much for your positive feedback.

Nevertheless, I do have some serious problems.

Thanks to your and other reviewers' comments, we re-structured our paper and enriched it with further experiments: Previous draft was reporting 330 ABE variants evaluated on 3 datasets, whereas this version of the paper reports 2090 ABE variants evaluated on 19 datasets. Below is the list of your comments and our replies to them.

First, though we need to deal with a potential publication bias against negative results, the work still needs to be interesting.

That was also the concern of other reviewers and we are thankful for that comment, we should have stated our motivation more explicitly. In the new version of our paper, we included a whole section about our motivation (Section 2) after the Introduction.

There's an excellent discussion of these kind of issues in [1].

We are grateful for that reference. We went through all *5* short papers in that reference and positioned our paper in the context of negative results according to our summary of these papers. We devoted a whole section (Section 3) concerning this issue and its relation to our paper. Section 3 reads:

> While it would have been gratifying to have found a positive result (e.g. that some kernel method was very much better), it is important to report such negative results as well. A very thorough discussion on the value of negative results can be found in [9]. The fundamental question is whether a negative

result poses a positive knowledge. Positive knowledge is defined by Browman et al. to be the ability of being certain, not being either right or wrong [9]. However, not all certain conclusions are *knowledge* per se. Common concerns are:

i. is the topic/hypothesis plausible,
ii. are the experiments sound,
iii. do the results propose *"negative evidence"* or *"non-conclusive search"* and
iv. will the reported results be valuable to future research.

As for ***i.***, research on weighting methods in ABE is quite plausible, see weighting method proposed earlier by Mendes et al. [35,36]. In that respect, our evidence of negative results serve the purpose of guiding research away from conclusions (such as kernel weighting can improve ABE performance) that would otherwise seem reasonable [9].

When presenting negative evidence it is crucially important to have *sound and extensive* experimentation (condition ***ii.***). This report rigorously investigates kernel weighting on *19* datasets subject to *3* performance measures through appropriate statistical tests.

The idea behind condition ***iii.*** is that *"one should disvalue inconclusive results"* [9], i.e. negative conclusions are more meaningful than *uncertainty.* The kernel weighting experiments of this paper on a wide range of ABE variants are negative evidence to conclude that it does not improve ABE performance, thereby satisfying ***iii.*** Finally condition ***iv.*** questions the benefit of results to future research. After years of research, effort estimation still suffers from conclusion instability. For stable conclusions, retiring a considerable portion of search space is as important as the discovery of successful applications. The contribution of this work is through retirement of *2090* of ABE variants.

The situation we have is: "Another possibility is that a novel hypothesis occurs to a scientist. This scientist proceeds to test. his bright idea only to discover that it is mistaken. Might he or she publish these findings to ward off other scientists who might be tempted to pursue this dead end?" The answer revolves around how interesting the results to which I'd answer not that interesting. My suggestion is a short note of perhaps 20% of the length of this paper.

Thanks to your valuable feedback. As an aside, we do mention that we have been talking to Briand and Basili about a *fast abstracts* section of ESE containing very small papers that need not have (e.g.) extensive background notes. To date, they have not been responsive so we are stuck with the current norms in scientific publishing; i.e. as much as possible, each paper has to be a stand-alone unit.

Nevertheless, we have significantly pruned this paper, from 29 pages to 21. And that is with new experiments (MAR performance measure is included) and much more data (the dataset number is increased from 3 to 19) and corrections.

The second problem is the paper isn't very well written - it reads like an early draft as it's often note like or lacks articles etc etc (below I've identified a few typos but there are a lot more).

We are sorry for that, we corrected all the below and did our best to present you a much better draft this time.

More seriously it's just hard to follow and a lot of irrelevant information is included eg in the intro why discuss feature selection when the paper is about analogy weighting and selection.

We removed that claim together with other less-relevant explanations.

Nor I do get any sense of why analogy weighting is an important and hard problem.

We wrote a new section (Section 2) so as to explain the motivation behind tackling the problem of analogy weighting.

Section 2 is weak eg why have 2 different taxonomies of related work – one will do!

We completely agree, we integrated these 2 sub-sections into 1 and included new references.

The discussion and Eqn 2 (for inverse rank weighting) seems unnecessary – it's pretty obvious.

You are right, we removed the discussion as well as the equation.

Your discussion implies k >1 since k=1 → U-ABE equiv N-ABE

That is very true, thanks for reminding us to explicitly state that. The 4^{th} sentence of the second paragraph in Section 5.3 states:

> Note that $k > 1$ for $\forall k$, because for $k = 1$ U-ABE and N-ABE would be equivalent.

p12 Much as I appreciate the work of Kitchenham and her co-workers this new definition of dataset quality seems desperately over-simplistic.

We dropped that claim.

MMRE? Given the authors are aware of its limitations why not do the statistical tests on absolute residuals? Why pred(25)?

Quite correct- we included absolute residual experiments in this version. However, we elected to *add* the one you suggest rather than *delete* the ones you depreciate since (as the reviewer might appreciate), other commenters on this paper might complain "why did they not use standard measures in wide use in the literature"? As for Pred(25) we made a small note in Section 5.4:

> Another alternative performance measure is PRED(25), which is reported to be one of the most commonly used accuracy statistics [26].

Minor points: p4, l36: problem explicitly addressed by Delphi where individual estimates are kept anonymous.

We removed that sentence.

p6, l22: a a

Corrected.

p6, final sentence: meaning is unclear.

We corrected this sentence as follows:

We want to remind researchers, who want to follow afore mentioned future directions that those future directions may as well end up in negative results.

p7, l27: statistical algorithms?
It is removed from the paper.

p9, l45: started
It is removed from the paper.

p17, l42: what were the imputed values (so the work could be replicated?)
The *"best-k"* approach is a pseudo-best k and it changes for different test instances, i.e. since we select ten random test instances and choose the lowest error yielding k value as the so called *"best"*, the values are somewhat random. We are repeating experiments 20 times for each setting and we are reporting that we do 20 runs to remove possible bias issues. Therefore, we reported our methodology instead of fixed *best-k* values, which might change for the next set of runs when another researcher is replicating our work.

p24, l37: does hardly let??
It is removed from the paper.

p26, l46: the choice of performance measure is not an open issue. See Kitchenham et al. [2]. The only issue is what characteristic of accuracy you are actually interested in, normally spread of residuals (error-proneness) and central tendency (bias) would be sufficient.
We dropped that claim.

REFERENCES: [1] H. Browman, "Negative results," Marine Ecology Progress Series, vol. 191, pp. 301-309, 1999. [2] B. A. Kitchenham, S. G. MacDonell, L. Pickard, and M. J. Shepperd, "What accuracy statistics really measure," IEE Proceedings - Software Engineering, vol. 148, pp. 81-85, 2001.

Thank you very much for these references, they were most helpful. We included them into the paper.