Using Unreliable Data for Creating More Reliable Online Learners

Leandro L. Minku and Xin Yao
Centre of Excellence for Research in Computational Intelligence and Applications
School of Computer Science, The University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK
Email: {L.L.Minku,X.Yao}@cs.bham.ac.uk

***Abstract*—Some machine learning applications involve the question of whether or not to use unreliable data for the learning. Previous work shows that learners trained using unreliable data in addition to reliable data present either similar or worse performance than learners trained solely on reliable data. Such learners frequently use unreliable data as if they were reliable and consider only the offline learning scenario. The present paper shows that it is possible to use unreliable data to improve the performance in online learning scenarios with a pre-existing set of unreliable data. We propose an approach called Dynamic Un+Reliable data learners (DUR) able to determine when unreliable data could be useful by maintaining a fixed size weighted memory of unreliable data learners. The weights represent how well learners perform for the current concept and are updated throughout DUR's lifetime. This approach manages not only to outperform an approach which uses only reliable data, but also an approach which uses unreliable data as if they were reliable. Moreover, the variance in performance is reduced in comparison to the approach which uses only reliable data. In other words, DUR is a more reliable learner.**

I. Introduction

Some machine learning applications involve the question of whether or not to use less reliable data, data with dubious usefulness or data which might be misleading. We will refer to such data as *unreliable* in this paper. For example, an application may involve whether or not to use data from several different companies when estimating software development effort for a particular company [1]; whether to use data from other (types of) buildings when predicting heat and electricity demand for a certain building [2], [3]; whether to use data from other organisations for fault-prediction [4]; etc.

Even though data from a particular organisation may be considered reliable by this organisation itself, it can be considered unreliable for another because of two main reasons. One of them is that a certain organisation's data may be of lower quality than another organisation's. As data collection frequently involves subjective issues, it cannot be guaranteed that two different organisations meet the same standards. The concept of what good quality is can vary from one organisation to another even if they are following the same guidelines for data collection. So, quality that is acceptable for one may not be for the other. For example, different software development companies may have different understandings of what is considered as high required software reliability.

The other reason is that even if data from a particular organisation have good quality, they may be only indirectly relevant, having dubious usefulness or even being misleading for another organisation. This happens because different organisations can be considerably different from each other. For example, a certain software development company may be used to deal with smaller size software projects than another, or a certain building may be located in a warmer area or have more or less sensors noise.

The examples above also illustrate the fact that the source of unreliability may affect not only labels, but also input attributes. So, techniques such as semi-supervised learning would not be able to directly deal with unreliable data.

Several studies in the literature conclude either that using unreliable data in addition to reliable data is detrimental to the performance or that at most similar performance to the sole use of more reliable data can be achieved. We will call comparatively more reliable data simply as *reliable*, for simplification. For example, Kitchenham, Mendes and Travassos [1] provide a review of approaches using cross-company data and within-company data for software effort estimation. There were no studies where cross-company models were significantly better than within-company models. Turhan et al. [4] also found that cross-company models were no better than within-company models in the context of defect prediction. Pedersena, Stangb and Ulsetha [3] and Cherkassky et al. [2] divide buildings in categories for electricity and/or heat demand prediction assuming that it is better to create separate models for each type of building due to their different characteristics.

When comparing models using mixed unreliable and reliable data to models using solely reliable data, existing studies frequently use unreliable data as if they were reliable [1]. In some cases, the worse performance provided by models using unreliable data inspired the use of filters to select only unreliable data believed to be more helpful [4], but even this way not achieving better results than using only reliable data. Existing studies also frequently analyse problems in a completely offline learning scenario, even though several potential applications actually operate in an online learning scenario, where more data can be acquired with time. For example, the true effort used to develop new software projects can be revealed with time, more heat and electricity demand values can be acquired with time, more data regarding the existence of faults are produced with time, etc.

The present paper aims at analysing if and how unreliable data could be used to *improve* the performance in comparison to using only reliable data and in comparison to using unreliable data as if they were reliable, considering applications which operate in an online learning scenario. We restrict our analysis to the case where there are pre-existing unreliable data available and only reliable data are continuously incoming. The case where unreliable data are also continuously incoming is left as future work.

We propose an approach called Dynamic Un+Reliable data learners (DUR) able to determine when unreliable data could be useful throughout the learning in order to improve performance. The main idea of the approach is that unreliable data can become more or less useful due to concept drift when considering online learning scenarios. Concept drift is a change in the underlying distribution of the data [5]. As an intuitive example, right after a drift, it may be better to use existing unreliable data that reflect somewhat the new concept while there are not enough reliable examples from the new concept. This idea is motivated by the benefit that very high diversity ensembles can bring to online learning. Such ensembles can become beneficial when a concept drift happens, even though they may be detrimental under a stable concept [5], [6].

Differently from previous works, DUR explicitly considers the relationship between benefiting from unreliable data and concept drift. It is able to improve performance by maintaining a fixed size memory of unreliable data learners associated to weights that represent how well they model the current concept. DUR manages not only to outperform an approach which uses only reliable data, but also an approach which uses unreliable data as if they were reliable. Moreover, the variance in performance is reduced in comparison to the approach which uses only reliable data. In other words, DUR uses unreliable data to compose a more reliable learning approach.

This paper is further organised as follows. Section II presents related work. Section III presents the proposed approach. Section IV presents the setup of the experiments considering the aim of the paper. Section V presents the experimental results and analysis. Section VI presents the conclusions and future work.

II. Related Work

The approach proposed in this paper is partially inspired on Dynamic Weight Majority (DWM) [7] (algorithm 1), which is a widely known ensemble approach for dealing with concept drift in online learning classification tasks. This approach maintains a set of base learners, each associated to a weight. Weights start with value one and are multiplied by a pre-defined parameter β, $0 \leq \beta < 1$, when their associated learner gives a wrong prediction in a time step multiple of p (line 9). DWM's predictions are based on the weighted majority vote among the base learners.

DWM also allows removal and addition of base learners at every p time steps (lines 16 and 19). Weights of new learners are initialised with value of one. The removal of learners is controlled by a pre-defined weight threshold parameter θ. In this way, new learners can be created to learn new concepts and poorly performing learners, which possibly learnt old concepts, can be removed. The algorithm normalises the weights by uniformly scaling them such that the highest weight will be equal to one. This is done to prevent newly added base learners from dominating the decision making of existing ones.

Algorithm 1 DWM [7]

Parameters:
p: period between learner removal, creation and weight update
β: factor for decreasing weights $0 \leq \beta < 1$
θ: threshold for deleting learners

```
1: $m = 1$ {Number of learners.}
2: Create learner $L_m$
3: $w_m = 1$ {Learner's weight.}
4: $t = 1$
5: for each new incoming data example $d = (\vec{x}, y)$ do
6:     $\vec{\sigma} = 0$ {Sum of weighted predictions for each class.}
7:     for each learner $L_i, 1 \leq i \leq m$ do
8:         if the estimation $\hat{y} = L_i(\vec{x})$ is wrong and
           $t$ mod $p == 0$ then
9:             $w_i = \beta w_i$
10:        end if
11:        $\sigma_{\hat{y}} = \sigma_{\hat{y}} + w_i$
12:    end for
13:    $\hat{y} = argmax_j \sigma_j$
14:    if $t$ mod $p == 0$ then
15:        Normalise all weights
16:        Remove learners $L_i \mid 1 \leq i \leq m, w_i < \theta$
17:        if the estimation $\hat{y}$ is wrong then
18:            $m = m + 1$
19:            Create new learner $L_m$
20:            $w_m = 1$
21:        end if
22:    end if
23:    for each learner $L_i, 1 \leq i \leq m$ do
24:        Use $L_i$ to learn $d$
25:    end for
26:    $t = t + 1$
27: end for
```

Additive Expert Ensemble (AddExp) [8] is an approach similar to DWM that works for both classification and regression tasks. For classification, AddExp learner's weights are multiplied by β, $0 \leq \beta < 1$, whenever the base learner gives a wrong prediction. This is similar to DWM using $p = 1$. For regression, weights are multiplied by $\beta^{|\hat{y}-y|}$, $0 \leq \beta < 1$, where $\hat{y}$ is the prediction given by the corresponding base learner and y is the actual value. This regression update rule only allows for predictions in the interval $[0, 1]$.

In our work, a weight update rule is also used to indicate what base learner is likely to be currently more helpful. As explained in section III, we propose a weight update rule that

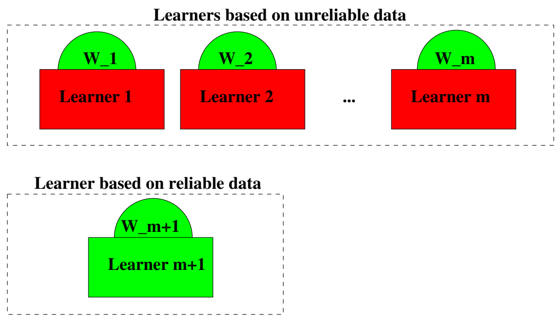

Learners based on unreliable data
W_1
Learner 1
W_2
Learner 2
...
W_m
Learner m
Learner based on reliable data
W_m+1
Learner m+1

Fig. 1. DUR's architecture. Components in green (light grey) are updated based on incoming reliable training data, whereas components in red (dark grey) are based solely on unreliable training data.

works for regression tasks with unbounded predictions.

The problem of using unreliable data is also related to inductive transfer learning. The latter regards to using knowledge acquired for solving a certain task in order to solve a different, but related, task [9]. For example, a system for classification of articles may recognise "news" articles more easily if it already learnt how to recognise "leisure". The problem considered in the present paper, on the other hand, considers exactly the same task, but using unreliable data to aid the learning.

III. Proposed Approach

We propose an approach called Dynamic Un+Reliable data learners (DUR) to use both unreliable and reliable data in an attempt to improve performance. The idea of the approach is that unreliable data may become more or less useful depending on the current concept. For example, a concept drift can make a certain organisation behave more similarly to another organisation. A learner based on unreliable data may perform better for a new concept than a learner based on an old concept or a learner trained with not enough data from the new concept. The literature on concept drift reports that learners that do not perform so well at a certain moment may become good in the future [5], [6].

With that in mind, DUR uses a fixed size memory of m learners trained on pre-existing available unreliable data and one learner specific for reliable data stream learning. Each learner is associated to a weight which represents how useful it currently is. These weights allow DUR to emphasize estimations given by unreliable data learners when they are useful, for instance while reliable data learners are still not adapted to a new concept. Figure 1 shows the proposed approach' architecture. DUR's predictions are based on the weighted average (for regression) or weighted majority vote (for classification) of the base learner's predictions.

Pre-existing unreliable data are separated into m different sets based on a priori knowledge. For example, one set can be created for each different organisation, or different sets can be created based on different ranges of a certain attribute/dependent variable. Separation based on ranges can be particularly useful, as they simulate different possible environments.

Algorithm 2 presents the learning algorithm. Learning is divided into two stages: offline learning based on unreliable

Algorithm 2 DUR

```
Parameters:
$Du_d, 1 \leq d \leq m$: unreliable data sets.
$\beta_u, \beta_r$: factors for decreasing learner weights $0 \leq \beta_u, \beta_r < 1$

 1: {Unreliable data learning:}
 2: for each unreliable data set $Du_d, 1 \leq d \leq m$ do
 3:     Create unreliable data learner $L_d$ using $Du_d$
 4:     $w_d = 1/m$ {Initialise weight.}
 5: end for
 6: $w_{m+1} = 0$ {Reliable data learner weight.}
 7: {Reliable data learning:}
 8: for each new reliable data example $d = (\vec{x}, y)$ do
 9:     for each unreliable data learner $L_i, 1 \leq i \leq m$ do
10:         if the estimation $\hat{y} = L_i(\vec{x})$ is "wrong" then
11:             $w_i = \beta_u w_i$
12:         end if
13:     end for
14:     if $d$ is the first reliable training example then
15:         $w_{m+1} = \frac{1}{m+1}$
16:     else
17:         if the estimation $\hat{y} = L_{m+1}(\vec{x})$ is "wrong" then
18:             $w_{m+1} = \beta_r w_{m+1}$
19:         end if
20:     end if
21:     Divide each weight by the sum of all weights
22:     Use reliable data learner $L_{m+1}$ to learn $d$
23: end for
```

pre-existing training data and online learning based on reliable training data stream. In the first stage, one base learner is created to learn each unreliable data set in an offline way (line 3). As future work, the case in which unreliable data are also incoming will be investigated. Weights associated to unreliable data learners are initialized to $1/m$ (line 4). Weights associated to reliable data learners are initialized to zero (line 6), so that DUR can provide predictions before reliable training examples become available.

The second stage of the learning is lifelong and one reliable training example from the stream is received at each iteration. Each reliable training example is used for the following:

1) Weights update (lines 9–21): each base learner is used to perform an estimation for the incoming training example. If the estimation is "wrong", the weight associated to the learner is multiplied by $\beta, 0 \leq \beta < 1$, where $\beta = \beta_u$ for the unreliable data learners and $\beta = \beta_r$ for the reliable data learners. Lower/higher β values cause the system to quickly/slowly reduce its emphasis on learners that are providing wrong estimations. The ideal values for β depend on the problem, but we suggest the median value of $\beta_u = \beta_r = 0.5$ as a default value.
For classification tasks, this rule behaves as DWM's [7]: an estimation is wrong when the wrong class is predicted. For regression tasks, the estimation is considered

wrong if the Magnitude of the Relative Error (MRE) [10] is higher than a constant $Pred$, where $MRE = \frac{|\hat{y}-y|}{y}$ for an estimated value $\hat{y}$ and an actual value y.
The weight update rule is used both for weights associated to unreliable and reliable data. In this way, weights are intended to represent how good each learner is for the current concept being learnt. When the first reliable training example becomes available, the weight associated to the reliable data learner is initialised as $1/(m+1)$. After all weights are updated, they are divided by the sum of all weights (line 21).

2) Reliable data learner's training (line 22): the incoming training example is learnt by the reliable data learner using its own learning algorithm. Unless the reliable data learner itself presents the ability to deal with concept drifts, DUR has no strategy to accelerate learning of new concepts by reliable data learners after drifts. Instead, estimations given by the unreliable data learners can be emphasized through their weights to keep relatively good performance while reliable data learners try to adapt to new concepts. The proposal of an additional strategy to accelerate adaptation of reliable data learners when there is concept drift is left as future work.

IV. Experimental Setup

As explained in section I, this paper aims at analysing *if* and *how* unreliable data could be used to improve the performance in comparison to models that use only reliable data and in comparison to models that use unreliable data as if they were reliable. The approach presented in section III represents a possible way (*how*) to improve the performance by using pre-existing unreliable data in online learning tasks. This section explains the experiments designed to evaluate the proposed approach to verify *if* it can really improve the performance.

A. Approaches Compared and Base Learners

The evaluation is based on comparisons of DUR against two other approaches. One of them consists in using only reliable data for the learning. This is done by using DWM (section II) to learn only the reliable data stream (DWM-R). The other one consists in treating unreliable data as if they were reliable. This is done by using DWM first to learn all the unreliable data as a stream and then to learn the reliable data stream (DWM-UR). DWM was chosen because it is a widely known concept drift online ensemble learning approach, besides having partially inspired our approach. In order to use it with regression tasks with unbounded predictions, we apply the same weight update rule for DUR and DWM.

The base learners were regression trees (RTs) using weka [11]'s REPTrees implementation. These RTs are more likely to produce good results in comparison to several other learning machines considering the data sets used in this study [12].

In order to understand the results obtained, the evaluation was also aided by the performance of an RT trained offline with each unreliable data set separately in comparison to an RT trained with the reliable data stream.

B. Performance Measure and Executions

The performance measure used in the analysis is the Mean Absolute Error (MAE) considering the predictions on the next ten examples of the reliable data stream. The performance is calculated at each position of the reliable data stream, which is called a *time step*. This measure was chosen because the data sets used in the evaluation represent regression tasks in which not many examples are produced per year. So, it is reasonable to request predictions on the next ten examples whenever a new example becomes available.

A single execution for each data set was performed for DUR and DWM-R, as they are deterministic when using the deterministic RTs from this study. As the order of presentation of unreliable data does not influence the results for DUR and DWM-R, but influences DWM-UR's results, and only the real chronological order for ISBSG is known, one DWM-UR execution was performed with the chronological order for ISBSG and thirty executions were performed with random unreliable data order for the other data sets.

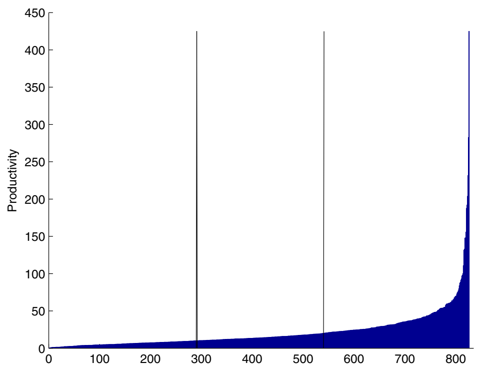
C. Data Sets

Three different data sets were used: ISBSG, CocNasaCoc81 and CocNasaCoc81Nasa93, as described below.

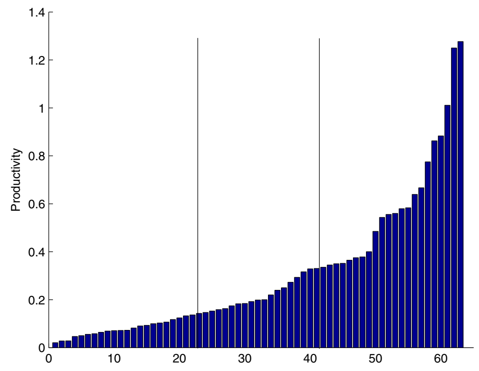
1) ISBSG: This data set was derived from the International Software Benchmarking Standards Group (ISBSG) Repository [13] Release 10, which contains information about software projects developed by several companies. We used the data for software development effort estimation and preprocessed them according to the ISBSG guidelines for that purpose. Four input attributes (development type, language type, development platform and functional size) and one output attribute (software effort in person-hours) were used. K-Nearest Neighbours [14] imputation was used for dealing with missing attributes.

The data set consists of 187 examples from a single company (considered here as reliable) and 826 examples from other companies (considered here as unreliable). As mentioned in section I, we consider data from other companies as unreliable because they may be only indirectly relevant and have unreliable quality. The usefulness of multi-company data for project estimation is questioned in the literature, with studies concluding that it produces similar or worse performance than single-company models [1]. Reliable examples were sorted according to the implementation date in order to compose the reliable data stream. Unreliable examples were also presented chronologically when their order of presentation influenced the results (DWM-UR). In order to produce different unreliable data sets for DUR, we separated the unreliable examples according to their productivity provided by the repository. The separation was based on the distribution of productivity (figure 2(a)) and is shown in table I(a).

2) CocNasaCoc81: Cocomo Nasa and Cocomo 81 are two software effort estimation data sets available from the PRedictOr Models In Software Engineering Software (PROMISE) Repository [15]. Both data sets contain 16 input attributes (analysis capability, programmers capability, lines of code, etc) and one output attribute (software effort in person-months).



(a) Unreliable ISBSG data



(b) Cocomo 81 data

Fig. 2. Distribution of Productivity

TABLE I
RANGES OF PRODUCTIVITY FOR CREATING SUBSETS OF UNRELIABLE ISBSG AND COCOMO 81 DATA

(a) Unreliable ISBSG data

Productivity Band	Number of Examples
[0,10]	291
(10,20]	250
(20,424.9]	285

(b) Cocomo 81 data

Productivity Band	Number of Examples
[0,0.15]	24
(0.15,0.35]	20
(0.35,1.30]	19

Cocomo 81 contains an additional input attribute (development type) not present in Cocomo Nasa, which was thus removed.

Cocomo Nasa's 60 examples were considered as the reliable data and Cocomo 81's 63 examples were considered as the unreliable data. The data sets provide no information on whether the projects are sorted in chronological order. The original order of the Cocomo Nasa projects was preserved in order to simulate the reliable data stream. The performance produced by an RT to learn the reliable data stream suggests that there is about one concept drift at approximately every fourth of this data set. Unreliable examples were presented in thirty random orders when their order of presentation influenced the results (DWM-UR). In order to create different unreliable data sets for DUR, the number of lines of code divided by the software effort in person-months ("productivity") was calculated. Its distribution is shown in figure 2(b). Examples were then separated according the ranges shown in table I(b).

3) CocNasaCoc81Nasa93: This data set is also composed of Cocomo Nasa and Cocomo 81, but it uses an additional data set called Nasa 93, which has the same input and output attributes as Cocomo Nasa and Cocomo 81. Even though both Cocomo Nasa and Nasa 93 are composed of Nasa's projects, they are used separately in the literature. For that reason, it is not known how useful Nasa 93 is for predicting Cocomo Nasa's projects. So, Nasa 93 was used here to compose an unreliable data set. Cocomo 81 was used as a single unreliable data set for DUR, instead of being divided into three. Similarly to CocNasaCoc81, the original order of the Cocomo Nasa projects was preserved in order to simulate the reliable data stream and unreliable examples were presented in thirty random orders when their order of presentation influenced the results (DWM-UR).

D. Parameters

Seven different combinations of parameters were used for DWM-R and DWM-UR: the default values of $\beta = 0.5$, $p = 1$ and $\theta = 0.01$, and all the combinations obtained by fixing two parameters to their default values and varying the remaining parameter using:

- $\beta \in \{0.3, 0.7\}$;
- $p \in \{10, 50\}$ for the larger data set ISBSG and $p \in \{10, 15\}$ for the other data sets; and
- $\theta \in \{0.001, 0.1\}$.

The parameters used for DUR were the default values of $\beta_r = \beta_u = 0.5$. In addition, as the data sets are regression tasks, the parameter $Pred$ needs to be set for all approaches. We chose $Pred = 25$ because estimations for these tasks are considered acceptable when $MRE < 25$ [16] and this value is widely used until nowadays (e.g., [17]).

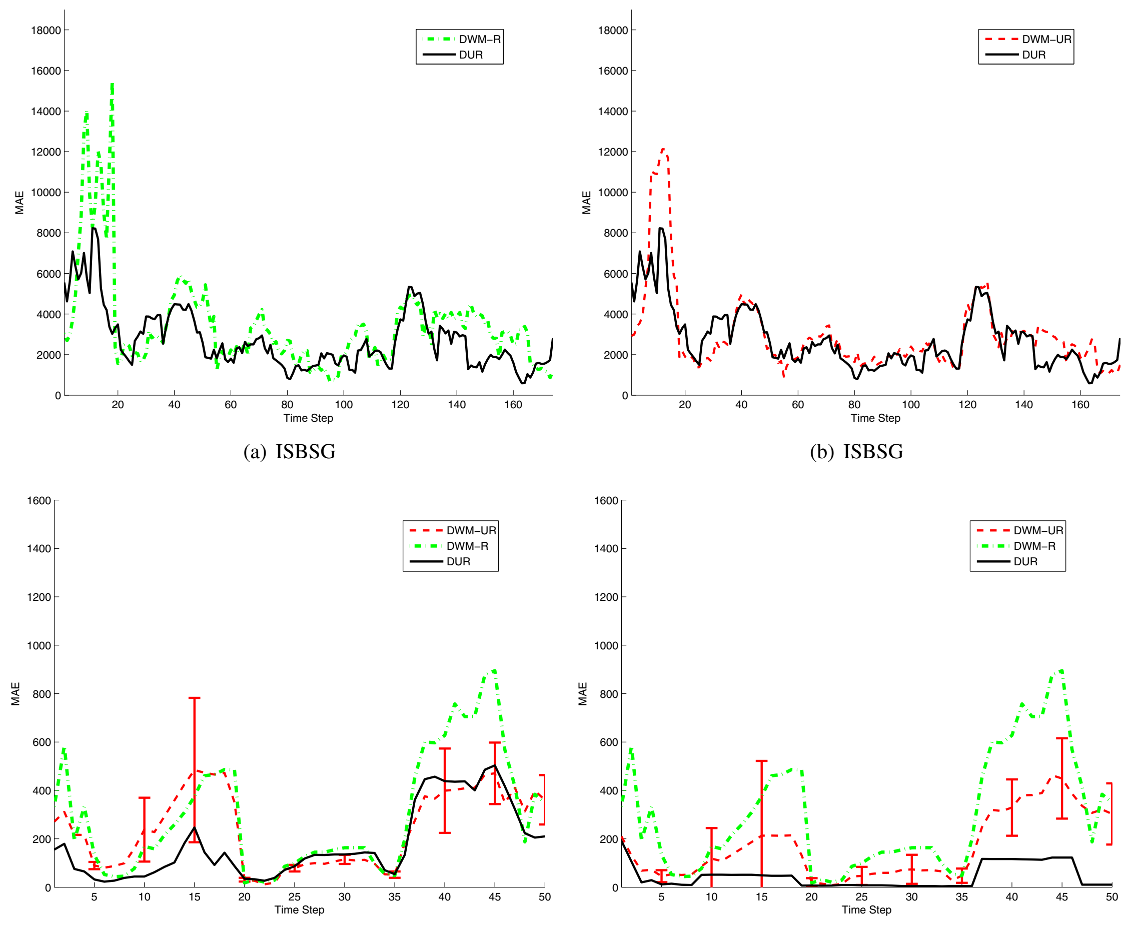
The parameters of the RTs were minimum total weight of one for the instances in a leaf, and minimum proportion 0.0001 of the variance on all the data that need to be present at a node in order for splitting to be performed. These parameters were the most likely to produce good results in [12].

V. EXPERIMENTAL RESULTS AND ANALYSIS

Sections V-A and V-B present the analysis of DUR in comparison to DWM-R and to DWM-UR, respectively.

A. DUR vs DWM-R

The performance obtained by DUR and DWM-R is shown in figures 3(a), 3(c) and 3(d). DWM-R's performance was better when using $\theta = 0.001$ for ISBSG and $p = 10$ for CocNasaCoc81 and CocNasaCoc81Nasa93 than when using the other parameters. So, only the performance using these parameters is shown. We can see that besides DUR's performance being better (lower MAE) than DWM-R's, it varies less across the time steps, being more reliable.



(a) ISBSG (b) ISBSG

(c) CocNasaCoc81 (d) CocNasaCoc81Nasa93

Fig. 3. DUR's performance in comparison to DWM-UR and DWM-R. The performances for ISBSG were separated into two figures to facilitate visualisation.

TABLE II
DUR'S, DWM-R'S AND DWM-UR'S PERFORMANCE STATISTICS CONSIDERING ALL TIME STEPS. STATISTICAL TESTS COMPARE DWM-R AND DWM-UR AGAINST DUR IN TERMS OF PERFORMANCE AVERAGE AND VARIANCE. P-VALUES LESS THAN 0.05/3 = 1.6667E-02 (MARKED WITH "*") INDICATE STATISTICALLY SIGNIFICANT DIFFERENCE AT THE OVERALL LEVEL OF SIGNIFICANCE OF 0.05 WHEN USING BONFERRONI CORRECTIONS CONSIDERING THE THREE DATA SETS.

Data Set	DUR	DWM-R		DWM-UR	
	Avg MAE +- Stdev	Avg MAE +- Stdev	Wilcoxon P-Value	Avg MAE +- Stdev	Wilcoxon P-Value
ISBSG	2,731.32 +- 1,516.38	3,477.15 +- 2,432.74	*3.9413e-08	2,995.90 +- 2,099.13	*4.2489e-03
CocNasaCoc81	175.8683 +- 147.2502	300.9366 +- 243.2639	*6.2792e-08	241.6329 +- 199.7075	2.8994e-02
CocNasaCoc81Nasa93	44.3515 +- 48.5396	300.9366 +- 243.2639	*7.5569e-10	165.0487 +- 185.3704	*9.2564e-12
	Variance	Variance	Levene P-Value	Variance	Levene P-Value
ISBSG	2,299,401.4982	5,918,245.6695	*1.3004e-02	4,406,377.8915	3.9226e-01
CocNasaCoc81	21,682.6078	59,177.3289	*9.8025e-05	39,883.0781	3.1777e-01
CocNasaCoc81Nasa93	2,356.0892	59,177.3289	*9.9920e-15	34,362.1682	2.7557e-02

DUR's and DWM-R's overall average performance and variance across all time steps further confirm this behaviour (table II). DUR's overall average performance was always better (lower MAE) and DUR's performance variance across time steps was always lower. The difference in average and variance is always statistically significant based on Wilcoxon sign rank [18] and Levene [19] tests, respectively, using Bonferroni corrections considering the three data sets and overall level of significance of 0.05 (statistically significant difference when p-value is less than 0.05/3 = 1.6667e-02).

In order to understand better how well DUR behaves, the performance of an RT trained offline with each unreliable data set separately before testing in comparison to an RT trained with the reliable data stream were analysed. A better performance of an RT trained on an unreliable data set indicates that this data set is beneficial. A worse performance indicates that it is detrimental. This analysis shows, for example, that for ISBSG one of the unreliable data sets is very beneficial from around time steps 5-20, and slightly beneficial after time step 145. Another unreliable data set is mostly detrimental,

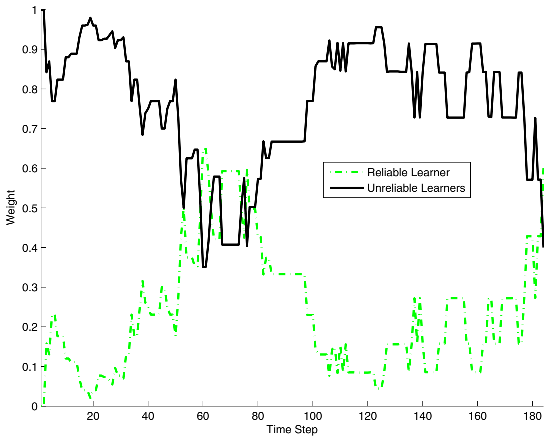


Fig. 4. DUR's reliable data learner weight and total unreliable data learner weight across time steps for ISBSG.

but beneficial from around 140-150. The remaining unreliable data set is mostly detrimental. We can see from figure 3(a) that DUR managed to benefit from the useful unreliable data sets, achieving better performance than DWM-R in particular during the periods in which they are beneficial. A similar behaviour was achieved for the other data sets, with DUR not benefiting from useful unreliable data only in one case where they were just slightly useful.

This analysis shows that DUR behaves well for improving the performance over DWM-R, usually benefiting from unreliable data sets when they are helpful without worsening performance when they are detrimental. Figure 4 shows as an example the weight given by DUR to the reliable data learner and the total weight given to the unreliable data learners across the time steps for ISBSG. We can see that unreliable data learners frequently contributed to more than 50 percent of the total weight. A high percentage was also frequently present for CocNasaCoc81 and CocNasaCoc81Nasa93 (omitted due to space restrictions). So, unreliable data play an important role in improving the performance.

In summary, DUR successfully detects when unreliable data sets are useful or detrimental, managing to improve performance and to reduce variance in comparison to an approach which uses only reliable data (DWM-R).

B. *DUR vs DWM-UR*

The performance of DUR and DWM-UR is shown in figures 3(b), 3(c) and 3(d). DWM-UR's performance using $p = 10$ was better, so the performance for other parameter values was omitted. Interestingly, the graphs show that DWM-UR's performance is sometimes better (lower MAE) than DWM-R's, revealing that using unreliable data as if they were reliable could be helpful in online learning scenarios.

And yet, DUR manages to achieve better performance than DWM-UR. Table II shows that DUR's overall performance considering all time steps together is better for ISBSG and CocNasaCoc81Nasa93 and the difference is statistically significant at the overall level of significance of 0.05 using Bonferroni corrections considering three data sets (statistically significant difference when p-value is less than 0.05/3). The statistical tests were Wilcoxon sign rank for ISBSG and Wilcoxon rank sum for CocNasaCoc81Nasa93, in which the sample size is different due to the thirty runs performed for DWM-UR. Even though there is no statistically significant difference for CocNasaCoc81 based on Wilcoxon rank sum with Bonferroni corrections considering all time steps together, there is statistically significant difference considering the first thirty time steps (p-value = 1.0709e-03), where figure 3(c) shows that DUR's performance is in general better. DUR's overall average MAE until time step thirty is 92.3663, whereas DWM-UR's is 202.3765.

TABLE III
NUMBER OF LEARNERS MAINTAINED ACROSS TIME STEPS FOR DWM-UR AND TOTAL WEIGHT FOR LEARNERS TRAINED WITH AT LEAST SOME UNRELIABLE DATA. THE WORD DATA IS ABBREVIATED AS "DT".

Time Step	Number of Learners Trained With		Un+Reliable Dt Learners Weight
	Reliable Dt Only	Un+Reliable Dt	
1-10	1	14	0.8739
11-20	2	13	0.7545
21-30	2	12	0.7368
31-40	3	12	0.6211
41-50	4	12	0.5411
51-60	4	10	0.5862
61-70	5	10	0.5806
71-80	6	10	0.4369
81-90	7	10	0.4516
91-100	7	10	0.4516
101-110	8	10	0.4000
111-120	8	9	0.4381
121-130	8	9	0.3962
131-140	8	7	0.3889
141-150	9	7	0.3398
151-160	9	6	0.2800
161-170	10	5	0.2039
171-180	9	5	0.1818
181-184	10	5	0.2264

DWM-UR's performance variance is decreased in relation to DWM-R's, not being statistically significantly different from DUR's. However, it is worth noting that DUR has the advantage of being independent of the order of presentation of unreliable data, whereas DWM-UR's performance can vary not only depending on the time step, but also on the order of presentation of unreliable data.

In order to provide a better understanding of the behaviours of the approaches, we further analyse the performance obtained for ISBSG, as we know the true chronological order of the unreliable data for DWM-UR's learning in this case. For this data set, DUR obtained better performance specially in the beginning of the time steps and slightly better in the end. During the mid time steps, the performance of these approaches was similar (figure 3(b)).

Table III shows the number of learners maintained by DWM-UR, created using solely reliable data and using at least some unreliable data. We will refer to these learners as reliable and unreliable data learners, respectively, for simplification. We can see that there is a large number of unreliable data learners during the first twenty time steps, when such data can be beneficial. However, DWM-UR's weight update at every $p = 10$ time steps keeps the total weight of unreliable data learners as 0.7545 at time steps 11-20. Our approach increases the weight from 0.88 to 0.96 (figure 4) during that period. The separation of unreliable data in ranges may have also helped to create higher quality estimators for DUR.

In the last time steps, when unreliable data can be more helpful again, our approach keeps them with high weight (figure 4), whereas DWM-UR eventually loses many of its unreliable data learners and gives them a low total weight. So, DWM-UR probably obtained worse performance because of the weight update at intervals of $p = 10$ and the loss of important unreliable data learners. This indicates that our approach's fixed memory of unreliable data learners is beneficial.

It is worth noting that DWM-UR using all other parameter configurations but $\theta = 0.001$ obtained considerably worse performance for ISBSG. The performance using $\theta = 0.001$ was just slightly worse than when using $p = 10$. The number of unreliable data learners from time steps 1-20 when using $\theta = 0.001$ was 44-35. Even though this number is large, the total weight was not so large (about 0.55 from time steps 10-20). The reason for that is that $p = 1$ causes DWM-UR to create many new learners (number of reliable data learners increased from 10 to 18 from time steps 10-20). As new learners always start with the highest weight, the total reliable data learners weight becomes quickly high. During the last time steps, $p = 1$ causes DWM-UR to lose even more learners trained with unreliable data, maintaining only three of them.

A parameter $\theta = 0$ would cause DWM-UR never to delete any learner, avoiding the loss of learners trained with unreliable data which could become useful in the future. However, that would cause the number of learners to be very high, in particular the number of learners trained only with reliable data, if we consider lifelong learning.

In summary, this section shows that DUR can improve the performance in comparison to an approach which uses unreliable data as it they were reliable (DWM-UR). Moreover, DUR is independent of the order of presentation of the unreliable data if the base learner used is also independent.

VI. Conclusions

This paper shows that it is possible to use unreliable data to improve the performance in comparison to a model that uses only reliable data (DWM-R) for online learning scenarios in which there is a pre-existing set of unreliable data. Differently from other works in the literature which consider completely offline learning scenarios, we show that the performance can be improved both by using unreliable data as if they were reliable (DWM-UR) and using an approach which we specifically developed for benefiting from unreliable data (DUR). Our proposed approach DUR presents better performance than both DWM-R and DWM-UR and is independent of the order of presentation of the pre-existing unreliable data. Moreover, the variance in performance is reduced in comparison to the approach which uses only reliable data. So, DUR uses unreliable data to compose a more reliable approach.

As future work, DUR should be evaluated using other types of base learners and more data sets, including classification tasks and unreliable data with high level of missing attributes. The performance should also be evaluated when not only reliable data, but also unreliable data are incoming. The proposal of a mechanism to accelerate the reliable data learner's adaptation to concept drifts and a theoretical analysis of the approach are also left as future work.

Acknowledgment

The authors would like to thank the SEBASE project members, the Data Mining Reading Group, and especially Dr. Rami Bahsoon for the useful comments and discussion. This work was supported by EPSRC grant (No. EP/D052785/1) and European Commission through its FP7 grant (No. 270428).

References

[1] B. Kitchenham, E. Mendes, and G. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE TSE*, vol. 33, no. 5, pp. 316–329, 2007.
[2] V. Cherkassky, S. Chowdhury, V. Landenberger, S. Tewari, and P. Bursch, "Prediction of electric power consumption for commercial buildings," in *IJCNN'11*, San Jose, CA, 2011, pp. 666–672.
[3] L. Pedersena, J. Stangb, and R. Ulsetha, "Load prediction method for heat and electricity demand in buildings for the purpose of planning for mixed energy distribution systems," *Energy and Buildings*, vol. 40, no. 2, pp. 1124–1134, 2008.
[4] B. Turhan, T. Menzies, A. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *ESE*, vol. 14, no. 5, pp. 540–578, 2009.
[5] L. L. Minku, A. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE TKDE*, vol. 22, no. 5, pp. 730–742, 2010.
[6] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE TKDE*, vol. 24, no. 4, pp. 619–633, 2012.
[7] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.
[8] ——, "Using additive expert ensembles to cope with concept drift," in *ACM ICML*, Bonn, Germany, 2005, pp. 449–456.
[9] R. Raina, A. Y. Ng, and D. Koller, "Contructing informative priors using transfer learning," in *ICML*, Pittsburgh, 2006, pp. 713–720.
[10] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *IEEE TSE*, vol. 32, no. 11, pp. 883–895, 2006.
[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.
[12] L. L. Minku and X. Yao, "A principled evaluation of ensembles of learning machines for software effort estimation," in *PROMISE*, Banff, Canada, 2011, p. 10p. [Online]. Available: http://dx.doi.org/10.1145/2020390.2020399
[13] ISBSG, "The International Software Benchmarking Standards Group." http://www.isbsg.org, 2011.
[14] M. Cartwright, M. Shepperd, and Q. Song, "Dealing with missing software project data," in *METRICS*, Sydney, 2003, pp. 154–165.
[15] J. Sayyad Shirabad and T. Menzies, "The PROMISE Repository of Software Engineering Databases." School of Information Technology and Engineering, University of Ottawa, Canada, http://promise.site.uottawa.ca/SERepository, 2005.
[16] S. Conte, H. Dunsmore, and V. Shen, *Software Engineering Metrics and Models*. Menlo Park, CA: Benjamin Cummings Publishing, 1986.
[17] E. Kocaguneli, T. Menzies, and J. Keung, "On the value of ensemble effort estimation," *IEEE TSE*, p. 14p. (in press), 2012.
[18] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, no. 6, pp. 80–83, 1945.
[19] H. Levene, *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, 1st ed. USA: Stanford University Press, 1960.