# On Dataset Complexity for Case Base Maintenance

Lisa Cummins and Derek Bridge

Department of Computer Science,
University College Cork,
Ireland
{l.cummins,d.bridge}@cs.ucc.ie

**Abstract.** We present what is, to the best of our knowledge, the first analysis that uses dataset complexity measures to evaluate case base editing algorithms. We select three different complexity measures and use them to evaluate eight case base editing algorithms. While we might expect the complexity of a case base to decrease, or stay the same, and the classification accuracy to increase, or stay the same, after maintenance, we find many counter-examples. In particular, we find that the RENN noise reduction algorithm may be over-simplifying class boundaries.

## 1 Introduction

*Case base editing* is a form of *case base maintenance* in which algorithms use heuristics to select cases to delete from case bases. In the context of case-based classification, *noise reduction algorithms* seek to delete noisy cases, with the goal of increasing classification accuracy; *redundancy reduction algorithms* seek to delete redundant cases, with the goal of increasing retrieval efficiency while, as much as possible, not harming classification accuracy.[1]

Suppose that we are comparing case base maintenance algorithms. How do we measure their efficacy? Obviously, we want to measure the number of cases that the algorithms delete from the case base: other things being equal, the more that are deleted the better. But we also need to see the effect on the resulting classifier. Often researchers report the change in the classification accuracy (or error) measured before and after maintenance on a separate held-out test set. But two problems arise.

The first problem, as is apparent from the previous paragraph, is that case base maintenance is a multi-objective optimization problem. We cannot, in general, maximize both the number of cases deleted and the accuracy of the resulting case base. We can see this in an extreme form by imagining an algorithm that proposes deletion of all but one case: deletion has been maximized but a case-based classifier will now incorrectly predict the class of all target problems whose true class is different from the class of the case that remains in the case base.

[1] This paper is based upon work partially supported by the Science Foundation Ireland under Grant Number 05/RFP/CMS0019.

In our previous work, we proposed two strategies for investigating the trade-off between these two objectives [3]. One was to compute the Pareto front, i.e. those algorithms not dominated by any other algorithms. The other was to combine the percentage of cases deleted and the percentage accuracy of the case base after maintenance into a single number. We proposed to use their harmonic mean.

But evaluating maintenance algorithms by the accuracy of the resulting case base has a second, little-recognized problem. Many of the maintenance algorithms use a classifier to decide which cases to delete (e.g. [1, 5]). Assuming that the maintenance algorithm uses the same (or nearly the same) classifier as we use to measure accuracy pre- and post-maintenance, we have no independent measure of what the maintenance algorithm is achieving, and we run the risk that our maintenance algorithm is directly (or nearly directly) optimizing that which we measure. It would be useful to have independent measures that could be used to evaluate maintenance algorithms. In this paper, we investigate using *dataset complexity measures*.

Section 2 describes the maintenance algorithms that we will be evaluating. Section 3 reviews dataset complexity measures, and selects the three that we will use in this paper. Section 4 explains our experimental methodology. Then Sections 5, 6 and 7 present the results for noise reduction algorithms, redundancy reduction algorithms and composites that combine the two, respectively. Section 8 describes some related work, and Section 9 draws conclusions and presents ideas for future work. This, to the best of our knowledge, is the first paper to evaluate case base editing algorithms using dataset complexity measures. Our conclusions are therefore tentative ones. But we find reason to doubt the efficacy of the RENN noise reduction algorithm.

## 2 Case Base Editing Algorithms

The most common case-base maintenance algorithms are listed in Table 1.

As the Table shows, there are two types of case-base maintenance algorithm: those that delete noisy (or harmful) cases, and those that delete redundant cases. In practice, case-base maintenance algorithms are often composites, comprising a noise reduction phase followed by a redundancy reduction phase. For example, Brighton & Mellish's Iterative Case Filtering (ICF) algorithm comprises a noise-filtering phase using RENN followed by their own bespoke redundancy reduction phase [1]. So that we can separately evaluate the individual components of these composites, we have extracted the redundancy reduction phase of the composite algorithms, naming them in the table, and treating them as separate algorithms. For example, ICFR is our designation for the redundancy reduction phase of ICF, and we refer to ICF by the designation RENN$\rightarrow$ICFR to emphasize its composite nature.

It is well-known that the algorithms in Table 1 perform differently on different datasets (see, e.g., [1]): one size does not fit all in case base maintenance. This follows naturally from the fact that each of the atomic maintenance algorithms heuristically targets different types of cases to remove:

Table 1: Atomic case-base maintenance algorithms, and classic composites

| Name used in this paper | Name in the literature | Description |
|---|---|---|
| *Atomic noise reduction algorithms* | | |
| RENN | RENN | Repeated Edited Nearest Neighbour [17] |
| BBNR | BBNR | Blame-Based Noise Reduction [5] |
| *Atomic redundancy reduction algorithms* | | |
| ICFR | — | Redundancy reduction phase of ICF [1] |
| RCR | — | Redundancy reduction phase of RC [13] |
| CRR | CRR | Conservative Redundancy Reduction [5] |
| *Classic composite algorithms* | | |
| RENN→ICFR | ICF | Brighton & Mellish's Iterative Case Filtering [1] |
| RENN→RCR | RC | McKenna & Smyth's algorithm [13] |
| BBNR→CRR | CBE | Delany & Cunningham's Case-Base Editing algorithm [5] |

- RENN regards a case as noisy if it has a different class to the majority of its $k$ nearest neighbours [17].
- BNNR, by contrast, regards a case as noisy if it causes other cases to be mis-classified [5].
- ICFR and CRR both regard 'interior' cases, i.e. ones within clusters of same-class cases, as redundant ones, and both aim to retain cases on the boundaries between classes because these cases are important for classification accuracy.
  - ICFR removes cases that are solved by more cases than they themselves solve [1].
  - CRR removes cases that solve other cases [5].
- RCR, however, aims to retain a case if it is surrounded by many cases of the same class, while treating as redundant, and deleting, those that surround it [13].

In this paper, we want to further explore the effect of these algorithms by looking at changes they bring in the values of dataset complexity measures.

## 3 Measures for Maintenance

Ho & Basu survey dataset complexity measures in [9, 10]. Subsequently, Orriols-Puig et al. have made available DCoL, the Data Complexity Library, which is an open-source C++ implementation of 13 measures based on those in the Ho & Basu survey [14].[2] However, we have found problems with the definitions of several of these measures. We present revised definitions in [2]. These are the measures whose labels contain a prime, e.g. $F_2'$. In [2], we also describe 4 additional measures: one is from the Case-Based Reasoning (CBR) literature [7]; we define the other 3 from ideas presented in the CBR literature [12, 16]. Table 2

[2] `http://dcol.sourceforge.net/`

Table 2: The dataset complexity measures considered in this paper. (For formal definitions, see [2].)

| Measure | Description |
|---|---|
| *Measures of overlap of attribute values* | |
| $F_1$ | Maximum Fisher's Discriminant Ratio |
| $F_2'$ | Volume of Overlap Region |
| $F_3'$ | Maximum Attribute Efficiency |
| $F_4'$ | Collective Attribute Efficiency |
| *Measures of separability of classes* | |
| $N_1'$ | Fraction of Instances on a Boundary |
| $N_2$ | Ratio of Average Intra/Inter Class Distance |
| $N_3$ | Error Rate of a 1NN Classifier |
| $L_1$ | Minimized Sum of Error Distance of a Linear Classifier |
| $L_2$ | Training Error of a Linear Classifier |
| $C_1$ | Complexity Profile |
| $C_2$ | Similarity-Weighted Complexity Profile |
| $N_5'$ | Separability Emphasis Measure |
| *Measures of geometry, topology and density of manifolds* | |
| $L_3$ | Nonlinearity of a Linear Classifier |
| $N_4$ | Nonlinearity of a 1NN Classifier |
| $T_1'$ | Fraction of Maximum Covering Spheres |
| $T_2$ | Number of Instances per Attribute |
| $T_3$ | Dataset Competence |

summarizes all 17 measures. Ho & Basu placed each complexity measure into one of three categories [9, 10], and we have also shown these in the Table.

For the purposes of this paper, we select just three of the complexity measures, one from each of the categories. We try to select ones that will reveal the changes that the maintenance algorithms bring about, and ones that are computed in a manner that is, as much as possible, independent of the way the maintenance algorithms work and the way we measure changes in accuracy.

**Measures of overlap of attribute values** The measures in this category characterize the extent to which the attributes in a dataset discriminate between the class labels. An attribute that takes one value $v$ in a case that is labeled by class $c^+$ but takes another value $v', v' \neq v$, in another case that is labeled by class $c^-$ is less discriminatory than an attribute whose values align perfectly with the class labels.

In [2], we did not find these measures to be predictive of classifier accuracy. But we might expect them to be much more revealing of the effects of maintenance, especially of noise reduction. An incorrectly-labeled case may decrease the discriminatory power of one or more attributes.

We choose to use $F_3'$, which is our modified version of Ho & Basu's maximum attribute efficiency. The efficiency of an attribute is defined as the proportion of cases in the case base which have values for that attribute that do not fall into the overlap for that attribute. We define the overlap for an attribute as the set of its values that appear in differently labeled cases; in other words, if $a$'s value is $v$ in case $x$ whose class is $c^+$, and $a$'s value is $v$ in another case $x'$ whose class is $c^-$, then $v$ is in $a$'s overlap.

We do not use $F_1$ because we cannot apply it to datasets that include symbolic values or multi-class classification (where there are more than two class labels); and we avoid $F_2'$ because, in the experiments that we describe in section 4, we found that most of its values on most datasets, both pre- and post-maintenance, tend to be zero. $F_4'$ would be just as good as $F_3'$: there is not much to choose between them here.

**Measures of separability of classes** The measures in this category estimate the length and linearity of class boundaries. We found all of these measures to be predictive of classifier accuracy [2].

We exclude $L_1$ and $L_2$ because they cannot handle multi-class classification problems and symbolic-valued attributes. We exclude $N_5'$ because it is simply the product of $N_1'$ and $N_2$ and so produces values intermediate between the two.

$C_1$ and $C_2$ are new measures that we introduced in [2], where we found them to be the two most predictive of classifier accuracy. We based both of them on Massie et al.'s case complexity measure [12]. But, when we computed these measures pre- and post-maintenance, we found, overwhelmingly, that they disagreed with the other measures about how complexity had changed, and so we exclude them here.

This leaves $N_1'$, $N_2$, $N_3$. We exclude $N_3$: it is computed using a 1NN classifier, and therefore may not be independent enough of the maintenance algorithms. The other two measures are reasonably independent of the maintenance algorithms. $N_1'$ creates a minimum spanning tree (MST) and counts the proportion of cases in the MST that are connected to cases of a different class. In our version, we repeatedly shuffle and recompute the MST, and average the results, which allows for the fact that MSTs are not unique. $N_2$ is the ratio between the sum of the distances from each case to its nearest like neighbour and the sum of the distances from each case to its nearest unlike neighbour. We will choose $N_1'$ since we found it to be more predictive of classifier accuracy than $N_2$ [2].

### 3.1 Measures of geometry, topology and density of manifolds

In this category, it is easy to choose $T_1'$. We exclude $L_3$ and $N_4$ because they cannot be computed on datasets with symbolic-valued attributes. $T_2$ is simply the number of instances in the dataset divided by the number of attributes: since the former will decrease (or, at least, not increase) and the latter will not change after maintenance, this measure is giving little insight beyond what we learn already from recording the percentage of cases deleted, so we exclude it. As for $T_3$, this is one of the new measures that we introduced in [2]; it is based on competence groups and their coverage [16]. We did not find this new measure to be predictive of classifier accuracy, and we have not found it to be useful here either: for all case bases and all maintenance algorithms, it shows complexity increasing, rather than decreasing, after maintenance. The notions of competence groups and their coverage were never designed for use as a complexity measure, and it is therefore no criticism of the competence idea that we have not found it to be useful either here or in [2]. It is possible that taking more account of the number of competence groups, rather than their coverage, would yield a more useful measure. Investigating this is left to future work.

For each case $x$ in the case base, $T_1'$ computes its adherence subset. This subset contains $x$ itself and all its nearest neighbours up to, but excluding, the nearest unlike neighbour. $T_1'$ is then the number of adherence subsets, ignoring those that are contained within, or equal to, another. While this measure was not very predictive of classifier accuracy [2], it behaves as we would expect for maintenance, mostly showing a decrease in complexity after maintenance.

In summary, then, we will proceed to use $F_3'$, $N_1'$ and $T_1'$ in the rest of this paper.

## 4 Experimental Methodology

We took 25 datasets: 19 from the UCI repository [8]; plus the Breathalyser dataset [6]; and five email datasets [5]. For evaluation, we performed repeated holdout on each of the datasets. Each dataset was divided randomly into three splits: a 60% training set, a 20% test set, and a final 20% which was required for evaluation of other systems in our research (not reported in this paper) and hence was discarded here. We created 10 different splits of the data and we report all results as averages over the 10 splits.

We ran each of the maintenance algorithms in Table 1 on the training set and recorded the percentage of cases deleted. We also recorded the accuracy (percentage of cases correctly classified) on the held-out test set both before and after maintenance. Additionally, we computed the values of each of the 17 complexity measures on the case bases before and after maintenance. We normalized the values of the measure to the $[0, 100]$ range in order to make comparisons easier. We also had the problem that in some measures low values mean low complexity but in others high values mean low complexity. In this paper, when computing changes in complexity, we normalize further to arrange matters so that a positive change always means a lowering of complexity.

Although making a positive change mean a decrease in complexity is perhaps counter-intuitive, we do this for the benefit of graphs later in the paper. It means that moving upwards and rightwards in the graphs is a good thing, corresponding to decreases in complexity and increases in classification accuracy respectively. We show this in Figure 1. The upper-right quadrant shows that the maintenance algorithm has unambiguously improved accuracy and complexity. The top-left quadrant signifies lower complexity but also lower accuracy. This might occur if the maintenance algorithm over-simplifies class boundaries. The bottom-left quadrant signifies higher complexity and lower accuracy. There are any number of reasons that this might happen but one is the deletion of correctly-labeled

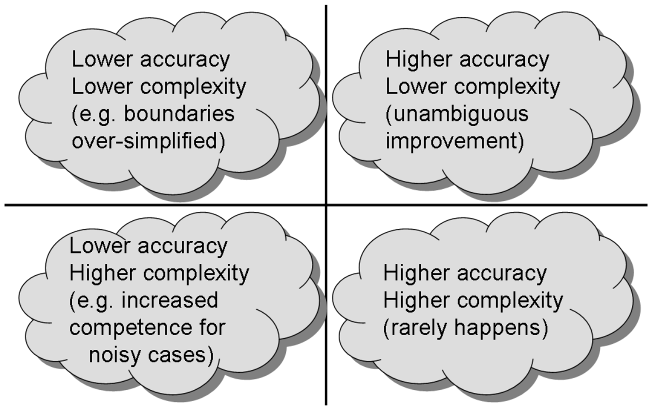


Fig. 1: Interpretation of graphs in this paper

Table 3: Percentage of cases deleted, change in accuracy, and change in normalized complexity measures: mean (and standard deviation) over 25 datasets. (In this paper, a positive change in a complexity measure implies lower complexity.)

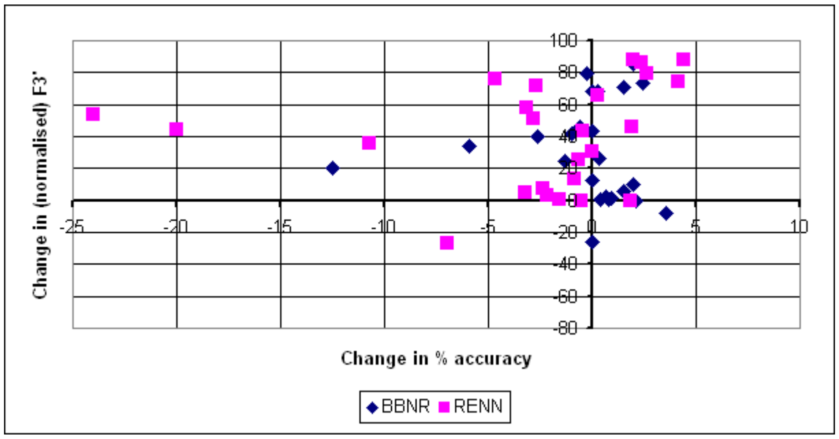
| | % of cases deleted | Change in % accuracy | Change in $F_3'$ | Change in $N_1'$ | Change in $T_1'$ |
|---|---|---|---|---|---|
| BBNR | 23.08 (15.06) | -0.24 (3.15) | 31 (31) | 6 (5) | 4 (5) |
| RENN | 23.17 (18.67) | -2.70 (6.74) | 41 (34) | 22 (18) | 23 (19) |
| CRR | 35.94 (7.86) | -2.00 (2.75) | 24 (32) | -14 (7) | -9 (6) |
| ICFR | 61.85 (22.15) | -4.97 (2.75) | 26 (34) | -37 (14) | -26 (15) |
| RCR | 77.59 (9.57) | -2.93 (2.43) | 32 (34) | -27 (9) | -14 (18) |
| BBNR→CRR | 59.38 (12.58) | -1.54 (3.37) | 44 (35) | 15 (17) | 16 (18) |
| RENN→ICFR | 78.41 (16.12) | -5.90 (6.64) | 48 (33) | -5 (26) | 10 (22) |
| RENN→RCR | 88.71 (2.11) | -4.25 (6.85) | 49 (34) | 1 (20) | 8 (24) |

cases near incorrectly-labeled ones, so that the incorrectly-labeled case will be retrieved for a larger number of target problems. The bottom-right quadrant is the situation of lower accuracy and higher complexity. Fortunately, our graphs show this to be very rare. Of course, as we said earlier, maintenance is a multi-objective optimization problem — losses in accuracy may be a price worth paying if the case base becomes more compact, and so it is not necessarily a 'bad thing' for a dataset and algorithm to be in a quadrant other than the top-right one.

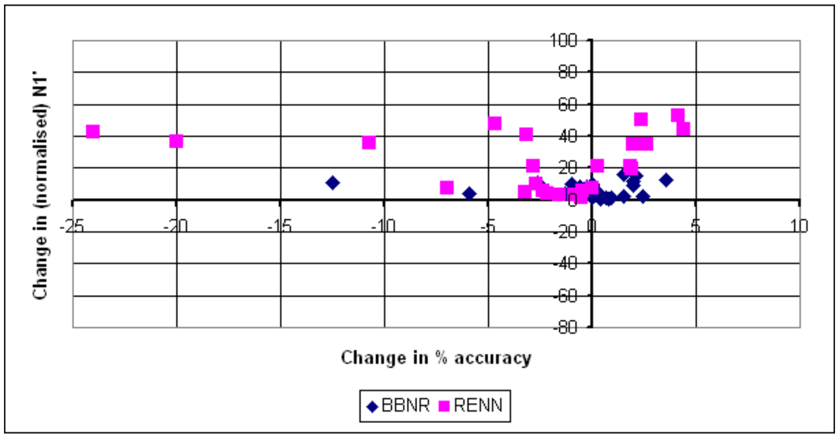
We summarize the results in Table 3, which shows averages over the 25 datasets, and reports the changes in the 3 measures that we selected in section 3.

## 5 Atomic Noise Reduction Algorithms

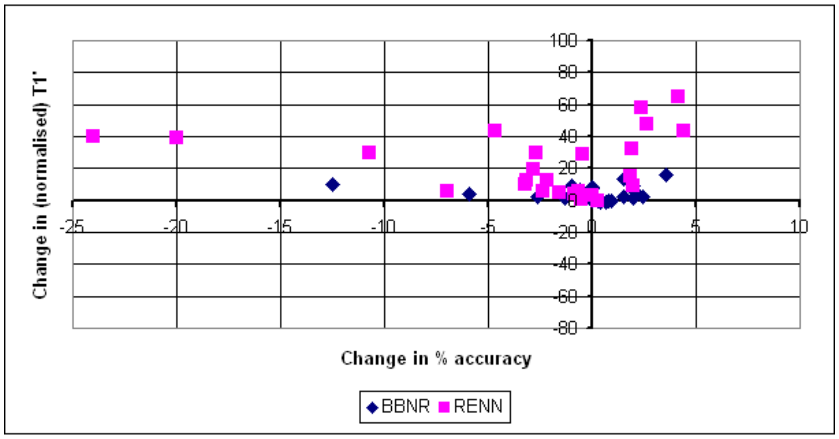
Figure 2 shows results for the atomic noise reduction algorithms, BBNR and RENN. The $x$-axis is the same in all three graphs in Figure 2. It is the change in



(a) $F_3'$: Maximum Attribute Efficiency



(b) $N_1'$: Fraction of Instances on a Boundary



(c) $T_1'$: Fraction of Maximum Covering Spheres

Fig. 2: Changes in accuracy and complexity: atomic noise reduction algorithms

accuracy on the held-out test set: positive values mean that accuracy is higher after maintenance than it was before maintenance. On the $y$-axis, we plot the change in the normalized values of the complexity measure: $F_3'$ in Figure 2a, $N_1'$ in Figure 2b, and $T_1'$ in Figure 2c. We repeat the point made earlier that, perhaps counter-intuitively, positive values here mean that the complexity has gone down after maintenance. Hence, the upper-right quadrant of every graph is where we might hope to be: increased accuracy and lower complexity. There are 50 points on each graph, representing what happens to each of the 25 datasets in the case of BBNR (diamonds) and in the case of RENN (squares).

We might expect that a good noise reduction algorithm would either improve accuracy or, if there was no noise, would leave it unchanged. But we see that the algorithms sometimes improve accuracy and sometimes worsen it. BBNR seems to do better than RENN: in 18 of the 25 datasets, BBNR improves accuracy or leaves it unchanged; in only 9 of the 25 does RENN do the same. In the case of RENN, some quite severe falls in accuracy occur (20-25%). We might think that the explanation would lie in the number of cases deleted. But, in fact, for most datasets, the two algorithms delete quite similar numbers of cases. The difference in the percentages deleted is less than or equal to 5% for 20 datasets. The mean percentage deleted over the 25 datasets by BBNR is 23.08%, where for RENN it is 23.17%. In only 7 of the datasets does RENN delete more, although in three of these the difference in the percentage deleted is more than 10%.

Both algorithms lower complexity, across all 3 complexity measures on nearly all datasets. Let's consider $F_3'$ first (Figure 2a). In an extreme example, if two cases have exactly the same attribute values but are labeled by different classes, then one of the cases is noisy (unless there are other attributes in the domain that would distinguish the two cases but which are not part of the dataset). We would hope that a noise reduction algorithm would delete at least one of these cases and ones like them and this should be reflected in an improvement to $F_3'$. Figure 2a shows that both algorithms do seem to be successful in doing this, and to the same degree. The average change in $F_3'$ in the case of BBNR is 31 with standard deviation (s.d.) also 31, and in the case of RENN it is 41 with s.d. 34.

A noise reduction algorithm might be expected also to 'clean up' the boundary between classes: removal of a noisy case should simplify the boundaries. This is what $N_1'$ is supposed to measure, and Figure 2b confirms that both algorithms succeed in improving this measure of complexity. RENN does this more aggressively than BBNR. Its improvements to $N_1'$ are often greater: it improves $N_1'$ by 22 on average (s.d. 18) whereas BBNR improves it by 6 (s.d. 5). But this may explain why RENN loses accuracy and therefore places many datasets in the upper-left quadrant: it may be over-simplifying class boundaries.

Finally, by removing noisy cases, a noise reduction algorithm should produce a case base in which clusters of like cases are fewer in number but larger in size. This is what $T_1'$ is supposed to measure, and Figure 2c shows again that this is what is happening: the changes are positive (meaning lower complexity), or close to zero, across all 25 datasets. Again RENN may be too aggressive: its mean is 23 (s.d. 19) compared to BBNR's mean of 4 (s.d. 5).

We conclude that, generally speaking, RENN's heuristic for identifying harmful cases is not as successful as BBNR's. It aggressively deletes cases, thereby over-simplifying the case base, achieving larger improvements in complexity but at the expense of classifier accuracy.

## 6 Atomic Redundancy Reduction Algorithms

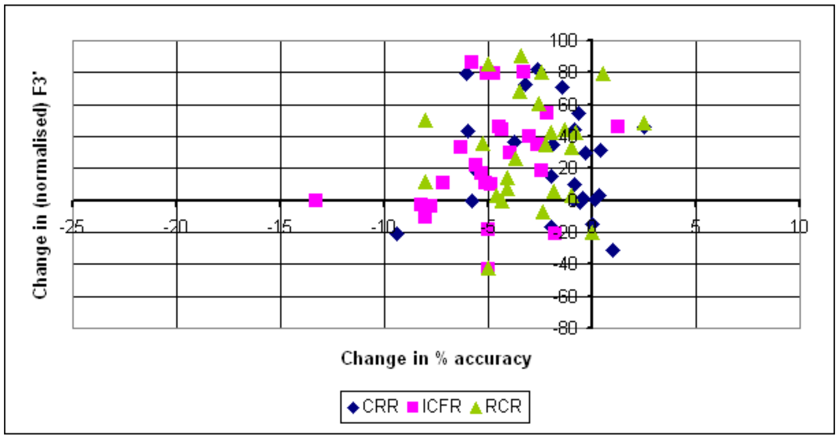
Figure 3 shows the performance of the three atomic redundancy reduction algorithms (CRR, ICFR and RCR). These algorithms are normally preceded by a noise reduction phase. It is important to be clear that here we are running these algorithms without previously doing any noise reduction. The algorithms may therefore not run as intended, because they usually expect to run on case bases with little or no noise, and the results will reflect this.

The standout observation across the three graphs in Figure 3 is that most points are to the left of the origin on the $x$-axis, showing that they mostly worsen classification accuracy. Of course, as explained above, this may be because the algorithms are not being applied to the kinds of case bases that they expect, i.e. ones that have undergone noise reduction. Another observation is that CRR (the diamond) tends to give smaller decreases in accuracy (mean -2%, s.d. 2.75) than RCR (the triangle, mean -2.93%, s.d. 2.43), which in turn gives smaller decreases than ICFR (the square, mean -4.97%, s.d. 2.75). This is only partly explained by looking at the percentage of cases deleted. CRR is conservative: it deletes the least and therefore does least damage to accuracy. Indeed it deletes 35.94% cases on average (s.d. 7.86). Counter-intuitively, though, RCR deletes the most (mean 77.59%, s.d. 9.57) with less damage to accuracy than ICFR, which deletes fewer cases on average (61.85%, s.d. 22.15).

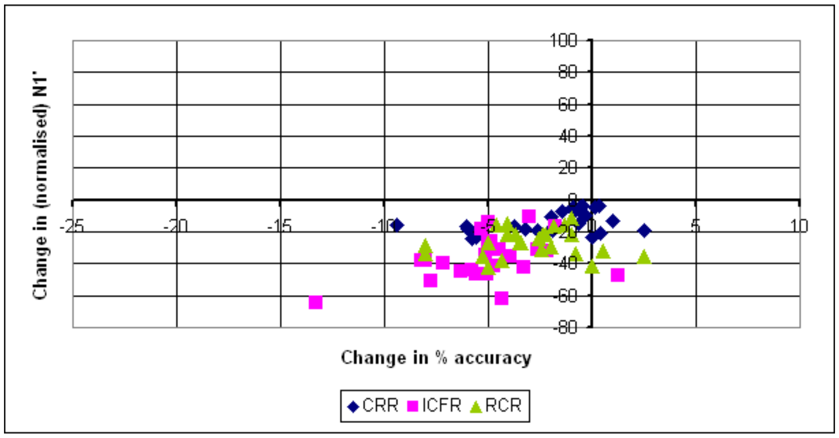
According to $F_3'$, complexity becomes lower (Figure 3a), but in the case of both $N_1'$ and $T_1'$, complexity increases (Figures 3b and 3c). In the case of $N_1'$ and $T_1'$, most points are in the lower-left quadrants. One hypothesis is that noisy cases, which have not previously been removed by a noise reduction phase, remain untouched by the redundancy reduction algorithms, and that correctly-labeled cases are removed. Without these correctly-labeled cases, the noisy cases grow in competence. They will be among the neighbours of a wider range of target problems, leading to reduced classifier accuracy. And, with fewer correctly-labeled cases relative to incorrectly-labeled ones, boundaries are more complex and there are more clusters, and denser clusters, of noisy cases.

## 7 Composite Algorithms

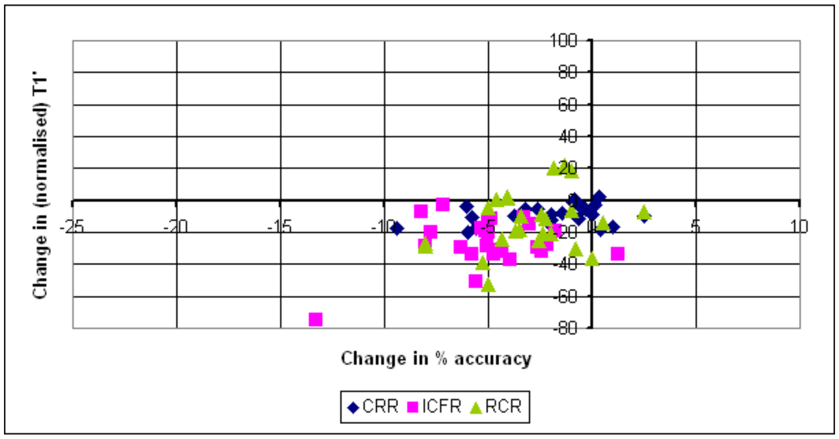
Figure 4 shows the effects of the composite algorithms, designated here by BBNR→CRR (diamond), RENN→ICFR (square) and RENN→RCR (triangle). We see a much greater spread in the effects on classifier accuracy: sometimes it improves; more often, it does not. BBNR→CRR does the least harm and has the narrowest spread (mean -1.54%, s.d. 3.37) compared to RENN→RCR (mean



(a) $F_3'$: Maximum Attribute Efficiency

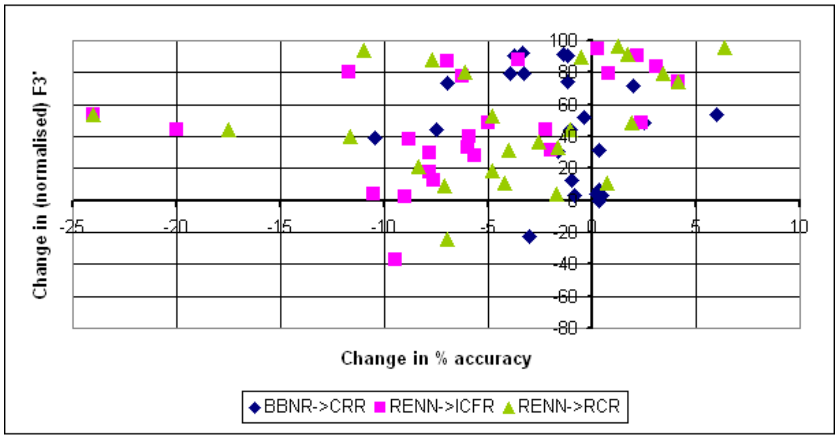


(b) $N_1'$: Fraction of Instances on a Boundary

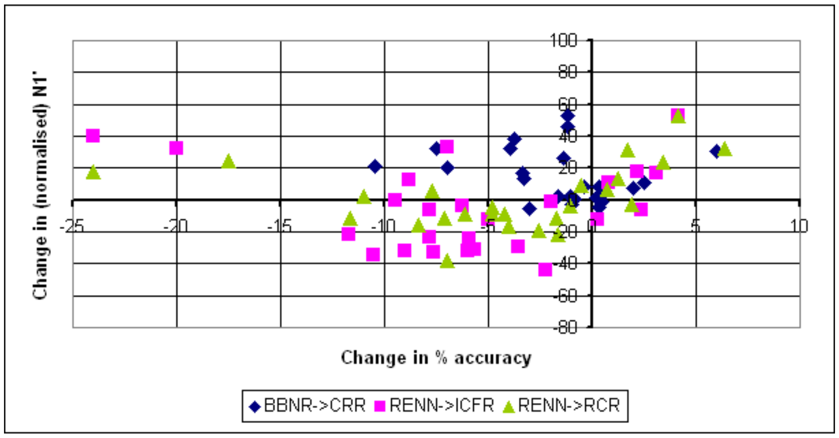


(c) $T_1'$: Fraction of Maximum Covering Spheres

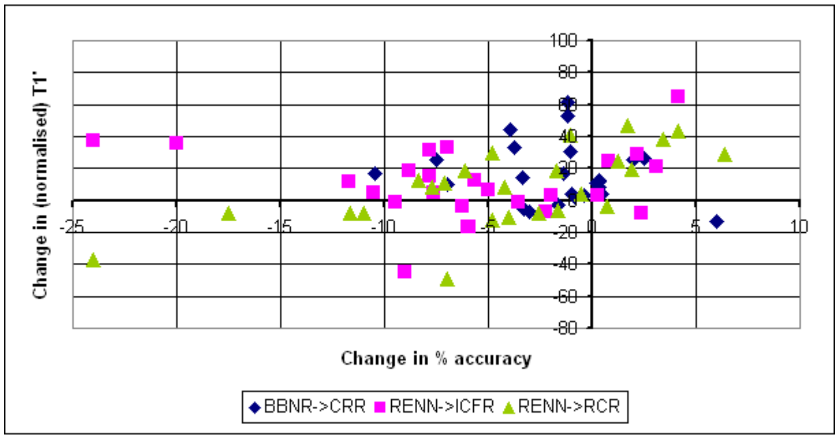
Fig. 3: Changes in accuracy and complexity: atomic redundancy reduction algorithms



(a) $F_3'$: Maximum Attribute Efficiency



(b) $N_1'$: Fraction of Instances on a Boundary



(c) $T_1'$: Fraction of Maximum Covering Spheres

Fig. 4: Changes in accuracy and complexity: composite algorithms

-4.25%, s.d. 6.85) and RENN→ICFR (mean -5.9%, s.d. 6.64), but it is also the ones that deletes the least — 59.8% compared to 78.41% and 88.71%.

Of course, these composites run redundancy reduction after noise reduction, and we find that the redundancy reduction is having a substantial effect. On its own, BBNR deletes 23.08% on average, but BBNR→CRR deletes 59.38% (36.3 more). RENN deletes 23.17% on average, but RENN→RCR deletes 88.71% (65.54 more) and RENN→ICFR deletes 78.41% (55.24 more).

The BBNR→CRR composite seems to have a clear advantage over its constituents, BBNR and CRR. BBNR→CRR deletes on average about the same amount as the sum of the amount deleted by BBNR alone and CRR alone. But the decrease in mean accuracy is less than the sum of the decreases caused by BBNR alone and CRR alone. In the cases of RENN→ICFR and RENN→RCR, the percentages of cases deleted by the composites exceed the amounts deleted by their constituents but, unless these very high levels of deletion are desirable, it might be better to use their redundancy constituents alone, since these still delete quite a lot of cases but with a smaller decrease in mean accuracy.

BBNR→CRR almost always lowers complexity: $F_3'$ improves by 44 (s.d. 35), $N_1'$ by 15 (17) and $T_1'$ by 16 (18). And it achieves these results with the least damage to accuracy. Results are more mixed for the other two composites. They both make improvements to $F_3'$ of nearly 50 (s.d. just over 30) and to $T_1'$ of about 10 (s.d. just over 20). But according to $N_1'$, RENN→ICFR makes complexity worse (-5 with s.d. 26) and RENN→RCR improves it but only by 1 (s.d. 20). Although BBNR→CRR and RENN→ICFR are more similar in their redundancy reduction phases (both aim to delete 'interior' cases), RENN→ICFR and RENN→RCR have the same initial noise reduction phase, and this seems to make their behaviour more similar in these results. Again, unless RENN's high levels of deletion are especially desirable, it may be better to avoid it in favour of BBNR→CRR, or just ICFR or RCR on their own.

In previous work, we briefly looked at the idea of composites in which ICFR and RCR were preceded by BBNR, rather than by RENN [3], and our new results here suggest that these should be investigated in more depth.

## 8 Related Work

There has been a number of studies that investigate the relationship between the dataset complexity measures and classifier accuracy, including [9–11]. But, we are presenting here the first ever study of complexity measures and case base editing algorithms.

The closest paper to our own is by Pranckeviciene et al. [15], which investigates a different form of maintenance, namely *dimensionality reduction*. They apply two dimensionality reduction techniques, Forward Feature Selection (FFS) and Linear Programming Support Vector Machine (LPSVM), to 5 high-dimension, sparse, bio-medical datasets. They choose 3 complexity measures to analyze the datasets before and after maintenance (using Ho & Basu's original definitions, not our modified ones): $N_1$, $N_2$, and $T_1$. The complexity results are somewhat mixed: complexity is sometimes reduced and sometimes not. The 3 measures that are used in [15] are selected to give insight into separability of classes, while being independent of classifiers. It might be interesting to choose a wider range of measures that provide different insights into the data.

## 9 Conclusions and Future Work

In this paper, we have reviewed a number of the most significant case base editing algorithms, along with a set of dataset complexity measures. We chose a small set of complexity measures, trying to ensure that they were different from each other (by choosing them from different categories of measures) and would reveal different insights into the operation of the maintenance algorithms, while at the same time preferring ones that were independent of classifiers (since some of the maintenance algorithms use classifiers in their operation). We then set up an experiment in which 8 maintenance algorithms were run on case bases created from 25 datasets. We reported the percentage of cases deleted, the change in accuracy, and the changes in the 3 complexity measures that we had selected.

This, to the best of our knowledge, is the first paper to evaluate case base editing algorithms using dataset complexity measures. Conclusions, at this stage, are only tentative, pending further research. In particular, we would like to use a wider range of datasets, e.g. image processing or bioinformatics datasets. Additionally, experiments with artificial datasets could be useful. They would allow us to control the degree of noise or possibly even the complexity.

In this paper, we have found reason to believe that the RENN noise reduction algorithm may be too aggressive. It often deletes a lot of cases but sometimes at the price of quite large decreases in classifier accuracy. At the same time, complexity often increases. This suggests that RENN may be over-simplifying class boundaries. Composite algorithms that use RENN as their initial phase are affected in a similar fashion. Indeed, the problems are compounded, whether the subsequent redundancy reduction phase targets interior cases (as with ICFR) or boundary cases (RCR).

The implications are that RENN needs much closer investigation. If it is important to delete a lot of cases, it may be better to use the ICFR or RCR redundancy reduction algorithms alone: fewer cases will be deleted, but the damage to accuracy may be lower.

We have found that BBNR may be doing a better job at identifying noise than RENN. It deletes as much on its own as RENN does, with less damage to accuracy. As suggested in [3], there is reason to investigate novel composites, in which BBNR precedes ICFR or RCR.

Finally, we think the ideas in [4] may be relevant to future work. In [4], Delany identifies 8 types of case profile, depending on the non-emptiness or otherwise of four sets associated with each case (the reachability set, the coverage set, the liability set and the dissimilarity set). Her paper looks empirically at what happens when cases with certain profiles are deleted. It would be interesting to see what effect deletion of cases with different profiles has on dataset complexity.

## References

1. H. Brighton and C. Mellish. On the consistency of information filters for lazy learning algorithms. In J. Rauch and J. M. Zytkow, editors, *Procs. of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pages 283–288, 1999.
2. L. Cummins. *Combining and Choosing Case Base Maintenance Algorithms*. PhD thesis, Department of Computer Science, University College Cork, Ireland, 2011 (forthcoming).
3. L. Cummins and D. Bridge. Maintenance by a committee of experts: The MACE approach to case-base maintenance. In L. McGinty and D. C. Wilson, editors, *Procs. of the 8th Intl. Conference on Case-Based Reasoning*, pages 120–134, 2009.
4. S. J. Delany. The good, the bad and the incorrectly classified: Profiling cases for case-base editing. In L. McGinty and D. C. Wilson, editors, *Procs. of the 8th Intl. Conference on Case-Based Reasoning*, pages 135–149, 2009.
5. S. J. Delany and P. Cunningham. An analysis of case-based editing in a spam filtering system. In P. Funk and P. González Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, pages 128–141, 2004.
6. D. Doyle, P. Cunningham, D. Bridge, and Y. Rahman. Explanation oriented retrieval. In P. Funk and P. González Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, pages 157–168, 2004.
7. A. Fornells, J. A. Recio-García, B. Díaz-Agudo, E. Golobardes, and E. Fornells. Integration of a methodology for cluster-based retreval in jcolibri. In L. McGinty and D. C. Wilson, editors, *Procs. of the 8th Intl. Conference on Case-Based Reasoning*, pages 418–433, 2009.
8. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
9. T. K. Ho and M. Basu. Measuring the complexity of classification problems. In *Procs. of the 15th Intl. Conference on Pattern Recognition*, pages 43–47, 2000.
10. T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.
11. N. Macià, E. Bernadó-Mansilla, and A. Orriols-Puig. On the dimensions of data complexity through synthetic data sets. In *Procs. of the 11th Intl. Conference of the Catalan Association for Artificial Intelligence*, pages 244–252, 2008.
12. S. Massie, S. Craw, and N. Wiratunga. Complexity profiling for informed case-base editing. In T. R. Roth-Berghofer, M. H. Göker, and H. A. Güvenir, editors, *Procs. of the 8th European Conference on Case-Based Reasoning*, pages 325–339, 2006.
13. E. McKenna and B. Smyth. Competence-guided case-base editing techniques. In E. Blanzieri and L. Portinale, editors, *Procs. of the 5th European Workshop on Case-Based Reasoning*, pages 186–197, 2000.
14. A. Orriols-Puig, N. Macià, E. Bernadó-Mansilla, and T. K. Ho. Documentation for the data complexity library in C++. Technical Report GRSI Report No. 2009001, Universitat Ramon Llull, 2009.
15. E. Pranckeviciene, T. K. Ho, and R. Somorjai. Class separability in spaces reduced by feature selection. In *Procs. of the 18th Intl. Conference on Pattern Recognition*, pages 254–257, 2006.
16. B. Smyth and E. McKenna. Modelling the competence of case-bases. In B. Smyth and P. Cunningham, editors, *Procs. of the 4th European Workshop on Case-Based Reasoning*, pages 208–220, 1998.
17. I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Trans. on Systems, Man, and Cybernetics*, 6(6):448–452, 1976.