2012

Profiling Instances in Noise Reduction

Sarah Jane Delany
Technological University Dublin, sarahjane.delany@tudublin.ie

Nicola Segata
Harvard University

Brian MacNamee
Technological University Dublin, brian.macnamee@tudublin.ie

Recommended Citation

Delany, S.J.,Segata,N. & Mac Namee, B. (2012) Profiling instances in noise reduction, *Knowledge Based Systems*, vol.31, p28-40. doi:10.1016/j.knosys.2012.01.015

Funder: SFI

Profiling Instances in Noise Reduction ☆

Sarah Jane Delany[a], Nicola Segata[b], Brian Mac Namee[c]

[a]*Digital Media Centre, Dublin Institute of Technology, Aungier Street, Dublin 2, Ireland*
[b]*School of Public Health, Harvard University, Huntington Ave, Boston MA, USA*
[c]*School of Computing, Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland*

Abstract

The dependency on the quality of the training data has led to significant work in noise reduction for instance-based learning algorithms. This paper presents an empirical evaluation of current noise reduction techniques, not just from the perspective of their comparative performance, but from the perspective of investigating the types of instances that they focus on for removal. A novel instance profiling technique known as RDCL profiling allows the structure of a training set to be analysed at the instance level categorising each instance based on modelling their local competence properties. This profiling approach offers the opportunity of investigating the types of instances removed by the noise reduction techniques that are currently in use in instance-based learning. The paper also considers the effect of removing instances with specific profiles from a dataset and shows that a very simple approach of removing instances that are misclassified by the training set and cause other instances in the dataset to be misclassified is an effective noise reduction technique.

Keywords: instance based learning, noise reduction, profiling, case-based editing

1. Introduction

Instance-based learning approaches such as Case-Based Reasoning rely heavily on the quality of the training data, and instance-based algorithms

☆This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/RFP/CMSF718.

such as k nearest neighbour are less noise tolerant than other machine learning techniques due to this reliance on individual training instances [1]. Research in this area has focussed considerably on developing training set editing techniques which can identify data instances that can be removed without affecting the competence of the training set. Much of this research was initially focussed on reducing the size of the training set without affecting the generalisation accuracy of the resulting classifiers [2, 3, 4]. As processing power increased, the measurement of performance moved from criteria measuring performance based on size to criteria measuring performance based on competence [5, 6, 7], and the more recent focus has been on the specific identification and removal of noise in datasets [8, 9, 10].

For a significant period of time the de-facto noise reduction approach in instance-based learning has been algorithms based on Wilson's Edited Nearest Neighbour (ENN) approach [11]. This is a simple technique that removes instances in a training dataset that do not agree with their nearest neighbours. Although there has been a significant amount of research into developing new editing techniques over the years, a lot of this work either develops a variation of ENN for noise reduction [12, 13] or incorporates a noise reduction pre-processing stage based on ENN or its variants [14, 5]. More recently there have been new editing techniques that focus specifically on the removal of noise or anomalous instances in the training set [8] typically by characterising the individual instances in the dataset to determine whether instances should be removed or not [15, 9].

In this paper we compare a number of noise reduction techniques that are in the current instance-based learning literature from the perspective of the types of instances they focus on for removal. This is done by analysing the structure of a training set at an instance level through the use of our novel instance profiling technique known as RDCL profiling [16]. Profiling the instances in a dataset allows for the categorisation of the instances based on their usefulness and the effect each instance has on the overall competence of the training set. Determining the different types of instances removed by the various noise reduction techniques can reveal the benefits and limitations of some of these techniques.

We also consider the removal of specific instance profiles from a training set and show that a very simple and conservative approach of removing instances that are misclassified by the training set and cause other instances in the dataset to be misclassified is an effective noise reduction technique.

Our comparison of noise reduction techniques in this paper is not just

with respect to the types of instances they focus on removing but is also a comprehensive comparison of performance. We compare their performance with respect to each other and with respect to how effective they are in improving the competence of an instance-based classifier.

The structure of the rest of this paper is as follows. Section 2 discusses existing work on editing strategies focussing on those that are designed to remove noise and improve competence. Section 3 describes the RDCL profiling technique used to allow the investigation of the structure of a dataset, and shows the effect of removing different categories of instances from a dataset. We show in this section how a very simple approach of removing instances that are misclassified by the training set and cause other instances in the dataset to be misclassified is an effective noise reduction technique. Section 4 then describes a comprehensive comparative evaluation of the noise reduction approaches using the profiling technique. This evaluation illustrates the types of instances that these techniques tend to remove which reveals the benefits and limitations of some of these techniques. Section 5 concludes the paper with directions for future work.

2. Related Work

Instance-based editing techniques have been categorised as focussing on *competence preservation* or *competence enhancement* [5]. Competence enhancement is effectively noise reduction, removing noisy or exceptional instances from the training set; while competence preservation can be considered redundancy reduction, removing superfluous instances that do not contribute to classification competence. Competence preservation techniques aim to remove internal instances in a cluster of instances of the same class and can predispose towards preserving noisy instances as exceptions or border instances. Competence enhancement on the other hand aims to remove noisy or corrupt instances but can remove exceptional or border instances which may not be distinguishable from true noise. A balance of both approaches can be useful in an instance editing algorithm.

Preprocessing the training set with noise reduction techniques also has theoretical motivations. It is desirable for the error of a classifier to be close to the Bayes error which is the theoretical minimum error for the optimal classifier on a particular training set. It is well-known that the classification error of a k nearest neighbour classifier (k-NN) with $k = 1$, is bounded by *twice* the Bayes error as the number of training examples n goes to infinity.

Devijver and Kittler [17] showed that it is possible to approach the Bayes error using a 1-NN classifier on a training set which was edited in such a way as that no misclassifications are possible within the training set itself. The k-NN algorithm, with $k > 1$, can also approach the Bayes error, but under the conditions that $n \to \infty$, $k \to \infty$ and $k/n \to 0$. These conditions are much more difficult to meet that those of the 1-NN classifier which only requires $n \to \infty$. Of course it is not possible to satisfy the $n \to \infty$ condition in practice, but a good noise reduction technique can help in approaching the Bayes error using non-infinite cardinality training sets.

There are good, comprehensive discussions and reviews of editing strategies covering both redundancy reduction and noise reduction, notably Brighton and Mellish [5] or Wilson and Martinez [18] which cover the earlier work well and more recently Segata et al. [8] which covers current work. As we are focussing on noise reduction this section will outline the main noise reduction techniques proposed and used in the literature on editing strategies. The earliest technique proposed for noise reduction was the Edited Nearest Neighbour technique [11] which, along with its variants [12, 13], has remained the most prominent technique in the literature. More recently proposed strategies include the Blame Based Noise Reduction approach introduced by Delany and Cunningham [15]; the Threshold Error Reduction algorithm suggested by Massie et al. [9] and the Local Support Vector Machine noise reduction approach offered by Segata et al. [8]. These techniques are discussed below.

2.1. Edited Nearest Neighbour (ENN) and its variants

The principle behind ENN is very simple–remove any instance that is misclassified by a k nearest neighbour (k-NN) classifier. This means that the class of the target instance as predicted by the classifier is not the same as its actual true class. Two extensions to this algorithm were proposed by Tomek [12]. The first is Repeated Edited Nearest Neighbour (RENN) which iteratively performs ENN on the training set until no more instances are removed. The second is the *all-kNN* algorithm which performs ENN multiple times on each instance in the training set starting with $k = 1$ and incrementing the value of k each time. Instances are removed if any value of k results in a misclassification.

RENN has in effect become the noise reduction technique of choice with a number of instance-based editing strategies including it as a pre-processing

pass to remove noise [14, 5, 18]. A more recent variation on ENN was proposed by Sánchez et al. [13] which uses a k nearest centroid neighbour (k-NCN) classifier rather than a nearest neighbour classifier. Nearest centroid neighbours are those that are the closest to the target but also are as symmetrically distributed around the target as possible [19].

This approach of removing misclassified instances from a training set is also used in work outside instance-based learning. Vezhnevets and Barinova [20] use a conceptually similar approach to avoiding overfitting in boosting by removing *confusing* instances, those instances that are misclassified by a perfect Bayesian classifier.

2.2. Blame Based Noise Reduction (BBNR)

BBNR [15] adopts an alternative principle to that used in ENN. Rather than focussing on the instances that are misclassified by the training set, it attempts to focus on those instances that *cause* other instances to be misclassified. BBNR identifies the training instances that are considered harmful by building a *competence model* of the training set. The competence model used derives from established case-base maintenance research [6] and identifies for each training instance, t, the other instances in the training set that t can classify correctly (captured as a set known as the *coverage set* [6]) and those instances that t causes to be incorrectly classified (captured as a set known as the *liability set* [15]). More details on the competence modelling technique are given in Section 3 as the RDCL profiling technique used in this paper also uses these basic sets. The coverage set effectively measures the potential 'good' associated with a training instance whereas the liability set measures the potential 'harm'.

The essence of the BBNR algorithm is to remove those instances in the training set that do more harm than good. The algorithm considers each harmful instance t, which are those with liability sets, and removes any t where the instances in the coverage set can still be classified correctly by the training set if t is removed.

A similar type of approach is used by Salamó and López-Sánchez [21] in ongoing case-base maintenance evolving the case-base by measuring the usefulness of cases for solving new problems using a 'goodness' measure inspired by re-inforcement learning.

BBNR was found to work effectively at editing training sets in the domain of spam filtering [15] but is not as effective in other text-based domains [22].

2.3. Threshold Error Reduction (TER)

The aim of the TER algorithm [9] is to identify and remove both noisy and harmful boundary instances. It calculates a *friend to enemy distance ratio* (F:E) value for each instance in the training set and uses thresholds on this value to determine which instances to remove. The F:E value of an instance t is the ratio of the *friend* distance (calculated as the average distance to t's three nearest neighbours of the same class) to the *enemy* distance (the average distance to t's three nearest neighbours of a different class). High F:E values indicate instances which are closer to instances of a different class and may potentially effect the generalisation accuracy of the training set. Instances, however, are only removed after a check to prevent removal of any instances that might appear to be noisy due to their noisy neighbours. As it is difficult to set a single threshold that works across all domains, the algorithm is iterative, reducing the threshold value at each iteration. It stops when the leave-one-out accuracy on the edited set is lower than that achieved on the previous iteration.

Massie et al. [9] also propose a simpler version of the algorithm, TER-Simple (TER-S), which simply takes a fixed threshold and removes all instances with F:E values higher than the threshold.

A significant limitation of the TER algorithm is that it is not viable for large datasets. Due to the leave-one-out validation that occurs at each step of the algorithm its performance can be unusably slow for large amounts of data. As an example of this, it took over 150 hours to run the TER algorithm on the largest dataset used in our evaluation which had over 6,500 instances, two classes and 166 features.

2.4. Local Support Vector Machine Noise Reduction (LSVM-NR)

The basis of the LSVM-NR algorithm [8] is to induce a localised SVM decision function around each training example and remove training examples for which the predicted probability of the actual class is below a specified threshold. In effect it removes an instance t that is too close to, or on the wrong side of, the maximal separating hyperplane which is built on the feature space projections of t's neighbouring instances. The technique is based on Local Support Vector Machines (LSVM) [23, 24] that consider locality explicitly and adopts the internal local model selection approach described in [25] to sensibly speed up and simplify the selection of the regularisation parameters and of kernel parameters.

Segata et al. [26] expand on this technique introducing FaLKNR, a fast and scalable noise reduction technique based on LSVM-NR, which is shown to be successful on very large training sets. The algorithm achieves scalability by reducing the number of local SVMs that need to be retrieved and built while ensuring the entire training set is covered by the models, and by using cover trees [27] to determine which instances to select as centres for the models.

3. Profiling Instances

In our earlier work [16] we introduced a profiling technique based on the local competencies of individual instances in a training set used in instance-based learning. The profiling technique allows the investigation of the effect that each instance in a training set has on the overall competence of the training set. This RDCL profiling technique builds on established case-base maintenance research [15, 16, 6] to model the competence of a training set which in turn allows the categorisation of each instance in the training set based on three characteristics;

(i) whether the instance is classified correctly or not by the rest of training set,
(ii) what benefit (or good), if any, it brings to the training set by its inclusion, and
(iii) whether or not it causes damage (or harm) to the training set by its inclusion.

Smyth and Keane [6] proposed two sets to model the local competence properties of an instance in instance-based learning; the *reachability* set of an instance t which is the set of all instances that can successfully classify t, and the *coverage set* of an instance t which is the set of all instances that t can classify. Using the training set T itself as representative of the target problem space, these sets can be estimated by definitions 1 and 2 below. Delany and Cunningham [15] extended this model to include an additional property; the *liability* set of an instance t which is the set of all instances that t causes to be misclassified and can be estimated by definition 3. Delany [16] further extended this model to include a final set, the *dissimilarity* set of an instance t which is the set of instances that contribute to the misclassification of instance t and can be estimated by definition 4.

$$
\begin{aligned}
\mathit{ReachabilitySet}(t \in T) &= \{t' \in T : \mathit{Classifies}(t', t)\} && (1)\\
\mathit{CoverageSet}(t \in T) &= \{t' \in T : \mathit{Classifies}(t, t')\} && (2)\\
\mathit{LiabilitySet}(t \in T) &= \{t' \in T : \mathit{Misclassifies}(t, t')\} && (3)\\
\mathit{DissimilaritySet}(t \in T) &= \{t' \in T : \mathit{Misclassifies}(t', t)\} && (4)
\end{aligned}
$$

where

$$
\begin{aligned}
\mathit{Classifies}(t', t) \text{ iff } t' \in \{\ & t'' \in T : t'' \in \text{ the } k\text{-neighbourhood of } t\\
& \wedge \text{ class of } t = \text{ class of } t''\\
& \wedge \text{ predicted class of } t = \text{ actual class of } t\}
\end{aligned}
$$

and

$$
\begin{aligned}
\mathit{Misclassifies}(t', t) \text{ iff } t' \in \{\ & t'' \in T : t'' \in \text{ the } k\text{-neighbourhood of } t\\
& \wedge \text{ class of } t \neq \text{ class of } t''\\
& \wedge \text{ predicted class of } t \neq \text{ actual class of } t\}
\end{aligned}
$$

In these definitions $\mathit{Classifies}(t, t')$ means that instance t' contributes to the correct classification of target instance t. This means that target instance t is successfully classified by an instance-based classifier such as k-NN and that instance t' is a nearest neighbour of instance t and has the same class as instance t. For k-NN with $k = 1$, instance t' 'causes' the correct classification as the closest instance to the target instance t but for $k > 1$ t', as just one of the closest instances to the target t, 'contributes' to the correct classification of t. In both scenarios k=1 and $k > 1$ instance t', as a neighbour of target instance t, will be a member of the reachability set of t and t will be a member of the coverage set of instance t'.

$\mathit{Misclassifies}(t, t')$ means that instance t' contributes in some way to the incorrect classification of target instance t. In effect this means that when target instance t is misclassified by the training set, instance t' is returned as a neighbour of t but has a different classification to t. Instance t is a member of the liability set of instance t' while t' is a member of the dissimilarity set of t. The dissimilarity set complements the reachability set in the same way that the liability set complements the coverage set.

Either the reachability set or the dissimilarity set for an instance t will always be empty. An instance t that has been classified correctly by the training set will, by definition, have a non-empty reachability set, $|RSet(t)| > 0$, and an empty dissimilarity set $|DSet(t)| = 0$. Vice versa for an instance t that has been misclassified by the training set, the dissimilarity set will be non-empty $|DSet(t)| > 0$, whereas the reachability set will be empty, $|RSet(t)| = 0$. However, the coverage and liability sets can independently be empty or non-empty, regardless of whether the instance is classified correctly or not. The coverage set of an instance t identifies the potential benefit or usefulness of t in the training set, represented by the instances that t contributes to classifying correctly. On the other hand the liability set of an instance t identifies the damage or harm that t causes in the training set represented by the instances that it causes to be misclassified. It is possible for an instance to be both useful for some targets and damaging for others.

The RDCL profile of an instance is derived by using the initial letter (R or D and/or C and/or L) of any of these four sets which are non-empty. An instance t can therefore have one of eight possible combinations of these letters with each combination or profile indicating (i) whether the instance is classified correctly or not, ($|RSet(t)| > 0$ or $|DSet(t)| > 0$); (ii) whether the instance is useful ($|CSet(t)| > 0$) and (iii) whether the instance is harmful or damaging ($|LSet(t)| > 0$).

Each of the eight possible profiles for instance t is defined and interpreted below.

R $|RSet(t)| > 0 \wedge |DSet(t)| = 0 \wedge |CSet(t)| = 0 \wedge |LSet(t)| = 0$
t is correctly classified but is not used for classifying any other instance in the training set.

D $|RSet(t)| = 0 \wedge |DSet(t)| > 0 \wedge |CSet(t)| = 0 \wedge |LSet(t)| = 0$
t is misclassified but is not used for classifying any other instance in the training set.

RC $|RSet(t)| > 0 \wedge |DSet(t)| = 0 \wedge |CSet(t)| > 0 \wedge |LSet(t)| = 0$
t is correctly classified and is useful in that it contributes to the correct classification of other instances in the training set.

RL $|RSet(t)| > 0 \wedge |DSet(t)| = 0 \wedge |CSet(t)| = 0 \wedge |LSet(t)| > 0$
t is correctly classified but is harmful causing damage by contributing to other instances being misclassified.

DC $|RSet(t)| = 0 \wedge |DSet(t)| > 0 \wedge |CSet(t)| > 0 \wedge |LSet(t)| = 0$
t is misclassified but is useful.

DL $|RSet(t)| = 0 \wedge |DSet(t)| > 0 \wedge |CSet(t)| = 0 \wedge |LSet(t)| > 0$
t is misclassified and causes harm.

RCL $|RSet(t)| > 0 \wedge |DSet(t)| = 0 \wedge |CSet(t)| > 0 \wedge |LSet(t)| > 0$
t is correctly classified and is both useful and harmful.

DCL $|RSet(t)| = 0 \wedge |DSet(t)| > 0 \wedge |CSet(t)| > 0 \wedge |LSet(t)| > 0$
t which is misclassified and is both useful and harmful.

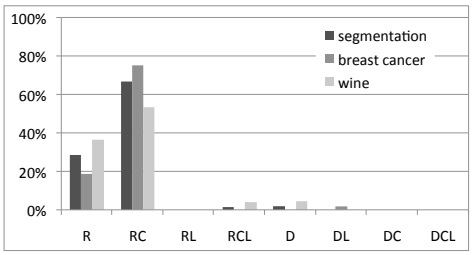
Figure 1 shows the profiles for a number of datasets, with Figure 1(a) including datasets that, in general, have good generalisation accuracy when a cross validation test is performed, and Figure 1(b) including datasets with poorer generalisation accuracy. The profiles are computed using $k = 1$, the effect of noise in the data is more evident with this value as higher values of k are more noise tolerant.

For datasets that separate well, it can be seen that the RC and R profiles are generally the majority instance profile types which is to be expected as these instance profiles cover instances that are classified correctly and contribute to the successful classification of others (in the case of RC instances). There are few instances with D or L in their profiles, which indicate instances that are misclassified and/or contribute to the misclassification of other instances. For datasets with poorer separability, the proportion of R and RC instances is considerably lower with the numbers of instances causing harm (with L in their profile) significantly higher.

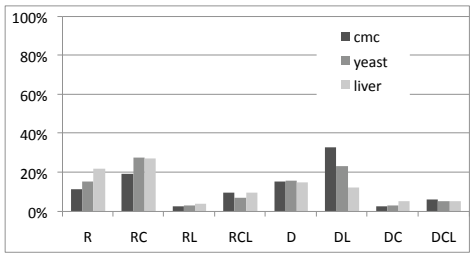
3.1. Visualisation of the training instances using profiles

To explore the structure of the datasets for which profiles are generated, and to better understand the effect of removing different kinds of instances, the datasets used in this paper have been visualised using a force-directed graph drawing algorithm visualisation technique [28]. This visualisation technique has been used before for, amongst other things, data exploration [29], similarity measure selection [30] and the visualisation of active learning approaches [31].

A force-directed graph drawing algorithm considers a dataset to be a maximally inter-connected graph in which each instance is represented as a node that is linked to every other instance (or node) in the dataset. An example is shown in Figure 2(a) in which instances a, b and c are shown as



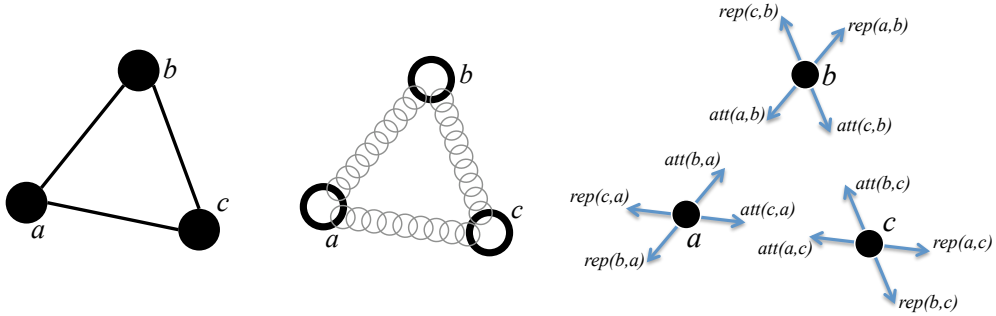
(a) Profiles of datasets with good cross validation generalisation accuracy (segmentation 96%, breast-cancer 95%, wine 95%)



(b) Profiles of datasets with poor cross validation generalisation accuracy (cmc 43%, yeast 49%, liver 61%)

Figure 1: Examples of dataset profiles illustrating the proportion of each type of instance. For details on the characteristics of the datasets, see Table 1.

a maximally inter-connected graph. The layout of the graph should be such that instances most similar to each other appear close together. To achieve this the metaphor of springs is used. Each instance is imagined as a steel ring with springs connecting it to every other instance (as illustrated in Figure 2(b)). The strength of the spring between any two instances is proportional to the similarity between the two instances, i.e. instances that are similar are linked by stronger springs than those linking instances that are dissimilar.



(a) (b) (c)

Figure 2: Dataset representations used by the force-directed graph drawing algorithm

To create a visualisation, the instances in a dataset are initially placed at random positions on a two-dimensional plane and are allowed to move according to the simulated spring forces until equilibrium is reached. As the system is allowed to find its own equilibrium the stronger springs will draw together those instances that are most similar to each other. So that all instances do not form a single small group, a repulsive force between each instance is also introduced.

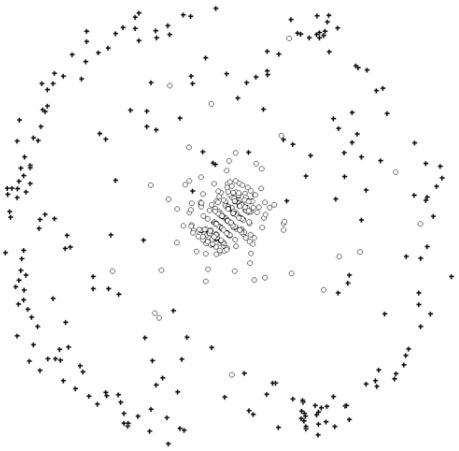
Figure 3(a) shows a visualisation of the breast-cancer dataset in which each instance is represented as a point in the graph and the colours and shapes of these points indicate their classes. As shown in this visualisation, this dataset is relatively well-behaved and the classes are very well separated (the cross validation accuracy for the data set shown in Figure 1(a) supports this). Figure 3(b) is a similar visualization of this dataset but displays the instances with R and RC profiles highlighted in black. This visualisation shows that the vast majority of instances are in the categories R and RC as was evident from the profile shown in Figure 1(a). Figure 3(c) shows that the small number of instances in the other *damaging* categories are scattered around the class boundary where there is less certainty with regard to classifications.

Figure 4(a) shows a visualisation of the liver dataset, again with classes indicated by shape and colour. This time it is clear that the class separation is not nearly as evident as was the case for the breast-cancer dataset. Again this is supported by the cross validation accuracy achieved on this dataset as listed in Figure 1(b). As this dataset is not easy to classify it is not surprising that the profiles are much more mixed than the breast-cancer dataset. Figure 4(b) highlights the instances with R and RC profiles, the useful instances, and we can see that there are fewer of these, relative to the breast cancer dataset, and they are distributed without the same correspondence to class positioning as in the breast cancer visualization above. Figure 4(c) shows the instances with profiles indicating damage, illustrating the larger proportion and wider distribution of these.

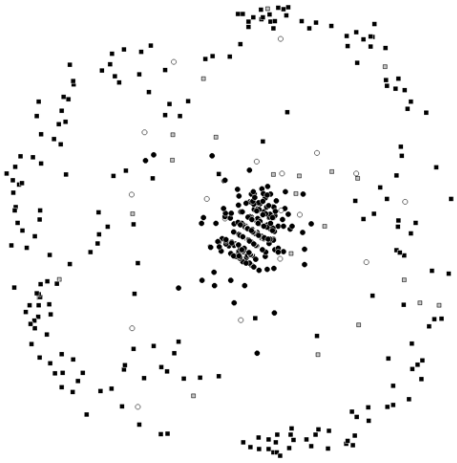
The effect of removing different types of instances will be shown in further visualisations in section 4.

3.2. The Effect of Removing Different Types of Instances

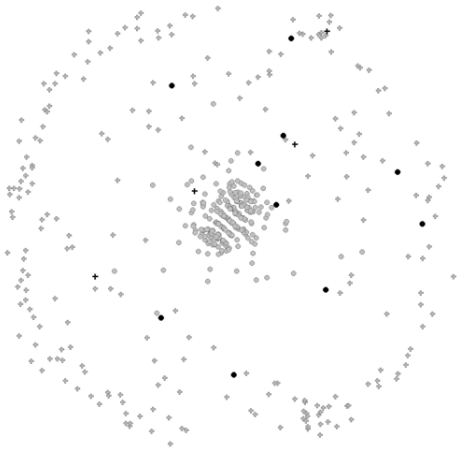
The ability to associate a competence profile with each instance in a dataset offers the opportunity to investigate the structure of datasets at an individual instance level and the effect of removing different types of instances



(a) Visualisation of the classes



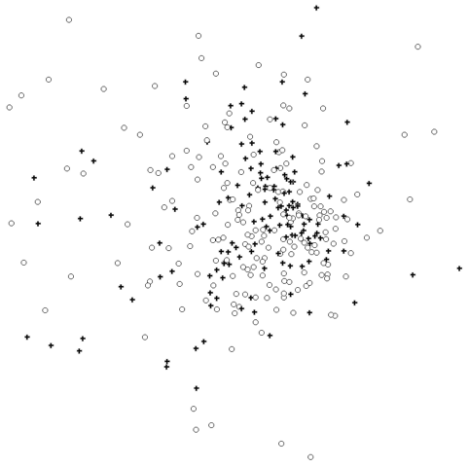
(b) Highlighting the R and RC profiles



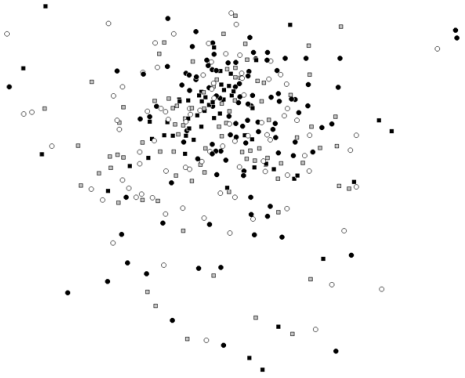
(c) Highlighting the DL profiles

Figure 3: A visualisation of the breastcancer dataset where (a) the two classes are indicated by the different colour and shape of the points and (b) the R and RC instances are highlighted in black and (c) the damaging DL instances are highlighted in black.

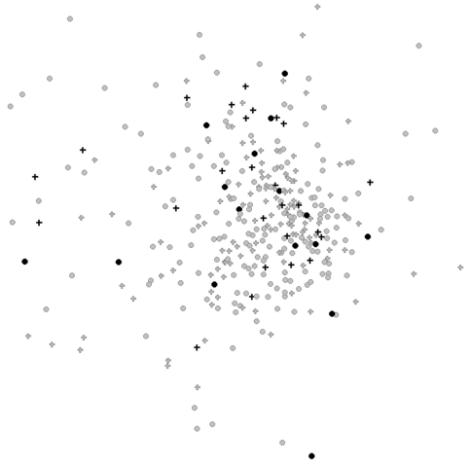
from the dataset. This section outlines a number of different investigations performed using profile information on a variety of datasets. Table 1 gives a list of the datasets used in this paper with a description of their characteristics. All datasets are available in the UCI repository [32]. Table 1



(a) Visualisation of the classes



(b) Highlighting the R and RC profiles



(c) Highlighting the DL profiles

Figure 4: A visualisation of the liver dataset where (a) the two classes are indicated by the different colour and shape of the points and (b) the R and RC instances are highlighted in black and (c) the damaging DL instances are highlighted in black.

also includes the baseline 10-fold cross validation accuracy achieved on each dataset using a k-NN classifier with a Euclidean distance measure. Since the objective is to consider the effect of different editing strategies, $k = 1$

Table 1: Datasets

	#instances	**#classes**	**#features**	**cv accuracy(%)**
australian	690	2	14	80.1
breastcancer	683	2	9	95.5
cmc	1473	3	9	43.3
diabetes	768	2	8	70.8
ecoli	336	8	7	81.0
glass	214	6	9	65.9
heart	270	2	13	75.6
hill-valley	1212	2	100	59.4
ionosphere	351	2	34	86.0
iris	150	3	4	95.3
liver	345	2	6	61.5
musk2	6598	2	166	73.8
segmentation	2310	7	19	96.3
sonar	208	2	60	61.1
spectf	267	2	44	71.5
vehicle	846	4	18	68.2
vowel	528	11	10	91.9
waveform	5000	3	21	77.2
wine	178	3	13	94.9
yeast	1484	10	8	48.8

was selected (the effect of noise in the data will be most evident with this value as higher values of k are more noise tolerant). Moreover, we already mentioned in Section 2 that applying the 1-NN classifier after preprocessing has theoretical motivations.

Table 2 shows the effect of removing all the instances of each profile type from each dataset. The accuracies were calculated using 10-fold cross validation, using the same folds as the original baseline unedited accuracy. A competence model was built on each training set of nine folds and the training set was edited to remove all instances with the specified profile. The

instances in the remaining fold were classified using the edited training set.

Table 2 also indicates whether removing a particular type of instance results in a significant (at the 5% level) improvement or dis-improvement in accuracy for a particular dataset. Significant decreases in accuracy are highlighted with a '–' next to the accuracy figure and significant increases highlighted with a '+'. The McNemar [33] test was used for testing for significance and where there were small levels of disagreement (i.e. ten or less) an exact sign test was used. Looking at individual dataset results suggests that removing RC or R instances causes significant decreases in generalisation accuracies, whereas removing DL instances causes significant improvements in generalisation accuracy. Using the more appropriate tests proposed by Demsar [34] for comparing two or more algorithms across multiple datasets supports these results. Using the Wilcoxon [35] Signed Rank test to compare the effect of removing each type of instance across all the datasets shows that removing R type instances and RC type instances has a significantly decremental effect on generalisation accuracy, whereas removing DL type instances has a significant improvement on generalisation accuracy.

Let us consider what this means.

RC and R instances:. RC instances are those that are classified correctly by the training set and are also used to correctly classify other instances. It would seem obvious that removing these types of instances would be detrimental to the generalisation accuracy of the training set which is what our evaluation shows. In fact Friedman's test (which tests whether the average ranks of a group of algorithms across multiple datasets are significantly different from the mean rank [36, 37]) followed by the Holm [38] post-hoc procedure shows that the pairwise comparison between the unedited dataset and the treatment of removing RC instances detects a significant difference – although this is the only difference that is detected as significant by this statistical test.

On the other hand instances with an R profile are instances that are classified correctly by the rest of the training set but are not used in the classification of any other instance in that training set. This might suggest that these are redundant and are not needed for generalisation but our evaluation shows a considerable dis-improvement in generalisation accuracy (with significance detected by the Wilcoxon test) when these are removed. This suggests that these may be outlier instances that are useful for generalisation across unseen data.

The instance profiles can also be computed with values for k higher than one and the results can be slightly different as each instance is more likely to be returned as one of the k nearest neighbours of other instances in the training set and thus contribute to the classification, correct or otherwise, of other instances. For $k > 1$, this means that the proportion of instances belonging to the R and D profiles will certainly decrease whereas the proportion of instances with RC, C, RCL and DCL profiles increase.

DL instances:. DL instances are those that are themselves misclassified by the training set and in addition cause damage by misclassifying other instances in the training set. It would again seem obvious that removing these would increase generalisation accuracy which we have shown is the effect.

RL and RCL instances:. Considering the average ranks of all treatments as shown in the second last row of Table 2 which according to Demsar [34] provide "a fair comparison of the algorithms", the removal of RL and RCL instances each ranks slightly higher than the accuracy on the baseline unedited dataset. The difference in average ranks is very small and without statistical significance which implies that removing these types of instances is unlikely to have a significant effect on the generalisation accuracy. RL and RCL instances are classified correctly by the training set but are causing harm overall by being instrumental in the misclassification of other instances. However, in addition, RCL instances are somewhat useful in that they contribute to the correct classification of other instances. This would indicate that these instances are located near the decision boundary and their removal may contribute to smoothing the decision boundary.

D, DC and DCL instances:. In contrast, the accuracy achieved by removing instances with a D, DC or DCL profile ranks lower overall (although again, not with statistical significance) than the accuracy on the baseline unedited accuracy. The differences in average ranks are again very small for the D and DCL instances, but considerably larger for the DC instances. Any instance with a D in its profile is misclassified by the training set but those with C and/or L are used to classify other instances. Again, this suggests that these instances are located at the decision boundary, but their removal overall is not necessarily advised by these results.

Instances with a D profile might, similar to those with an R profile, be considered to be redundant as they are not used in the classification of any other instance in the training set. However, in line with the argument for

Table 2: Accuracy values (%) on editing the datasets by removing instances with a specific instance profile. Values which show a significant decrease in accuracy over the unedited baseline using McNemar's test are highlighted by '–' next to the accuracy figure while accuracy values which show a significant increase are highlighted by '+'. The average rank of each algorithm is given in the second last row and the last row indicates whether the results of a Wilcoxon signed rank test between each algorithm and the unedited dataset is significant (Y) at the 5% level.

Dataset	**unedited**	**R**	**RC**	**RL**	**RCL**	**D**	**DC**	**DL**	**DCL**
australian	80.1	79.6	78.3–	79.9	83.4	79.6	80.6	83.5+	78.7-
breastcancer	95.5	95.5	94.4–	95.9	95.5	95.9	95.6	96.1	95.5
cmc	43.3	42.5	42.9	42.9	42.8	42.8	42.7	46.2+	44.1
diabetes	70.8	70.1	66.0–	70.6	69.7	71.4	71.0	74.1+	70.2
ecoli	81.0	78.9–	77.1–	80.3	81.0	82.1	80.1	83.3+	81.0
glass	65.9	65.9	62.2–	66.4	64.5	65.4	65.9	65.0	65.9
heart	75.6	76.7	72.2–	75.6	75.6	75.9	76.0	77.4	75.2
hill-valley	59.4	57.7–	59.2	59.7	59.7	53.1–	57.8–	59.5	59.1
ionosphere	86.0	87.2	81.5–	86.3	86.3	85.8	84.6	86.3	86.6
iris	95.3	95.3	95.3	95.3	95.3	96.0	95.3	95.3	95.3
liver	61.5	61.5	60.6	61.5	61.5	60.6	60.3	62.0	62.3
musk2	73.8	72.4–	70.5–	74.0+	74.5+	74.9+	73.8	74.9+	73.8
segmentation	96.3	95.8–	95.6–	96.2	96.3	95.6–	96.0–	96.1	96.2
sonar	61.1	58.2–	60.6	59.1	62.5	59.1	61.1	62.0	61.1
spectf	71.5	70.4	67.8–	71.9	72.7	72.7	70.8	73.4	71.5
vehicle	68.2	69.0	64.9–	67.9	68.4	68.6	68.1	68.2	68.0
vowel	91.9	90.2–	75.0–	92.4	91.9	82.4–	91.5	87.1–	91.9
waveform	77.2	76.7	74.5–	77.2	76.9	77.2	77.0	78.0+	77.6
wine	94.9	93.8	93.8	94.9	94.4	94.4	94.9	94.9	94.9
yeast	48.8	47.1–	46.2–	48.9	49.5	50.5+	48.9	50.1	48.5
Avg rank	4.5	6.3	8.3	4.2	4.3	4.7	5.5	2.8	4.7
Wilcoxon		Y	Y					Y	

the R profile instances discussed above, they may prove useful in classifying unseen examples. Interestingly, in spite of the evidence here suggesting keeping these instances, all D, DC and DCL instances are removed by the standard Wilson noise reduction technique used in many editing techniques which removes all instances that are misclassified by the training set.

4. Comparison of Noise Reduction Algorithms

The objective of this evaluation was to compare existing noise reduction algorithms and, using the profile framework, identify the types of instances that the algorithms focus on for removal. The noise reduction algorithms to be compared in this work include RENN, BBNR, LSVM-NR and TER which are all discussed in detail in Section 2 and are recently proposed noise reduction algorithms in the area of instance-based learning. Two versions of LSVM-NR were considered, the first using a linear kernel function denoted LSVM-lin, and the second using a radial-basis kernel function, LSVM-rbf. Two versions of TER were also considered, the non-iterative version TER Simple (denoted TER-S) which uses a F:E threshold of 1 in determining which instances to delete and the standard TER algorithm which uses a variable F:E threshold starting at 1.25 and reducing by 0.1 on each iteration stopping when the threshold reaches 0.5 if the main stopping criterion of increased leave-one-out accuracy was not already reached[1].

We also included in the comparison removing the DL instances from each dataset as we showed in the previous section that removing DL instances resulted in a significant improvement in generalisation accuracy.

The evaluation consisted of a 10-fold cross validation across all the datasets listed in Table 1 using the same folds for each algorithm. Each training set of nine folds was edited using the various noise reduction algorithms and the instances in the remaining fold were classified with a 1-NN classifier using the resulting edited training set. The results are presented in Table 3 which gives the accuracy achieved by each algorithm on each dataset. Table 3 also indicates whether the algorithms result in significant improvements or dis-improvements in accuracy when compared to the baseline unedited dataset using McNemar's test with significant decreases in accuracy highlighted with

[1]This additional stopping criterion was added after communications with the authors of the TER algorithm [9] as the algorithm failed to stop on some of the datasets in this evaluation

Table 3: Accuracy values (%) on editing the datasets using the different noise reduction algorithms. Values which show a significant increase in accuracy over the unedited baseline using McNemar's test are highlighted by + while accuracy values which show a significant decrease are highlighted by –. The percentage of datasets which show an overall increase or decrease are listed in the row labelled % +/–; with percentages of datasets with significant differences in the row labelled #sig +/– . The last row shows those algorithms marked Y which exhibit a significant difference over the unedited dataset using the Wilcoxon test.

Dataset	**unedited**	**LSVM -lin**	**LSVM -rbf**	**TER-S**	**TER**	**RENN**	**BBNR**	**DL**
australian	80.1	85.1+	84.9+	85.1+	85.6+	85.2+	83.8+	83.5+
breastcancer	95.5	97.1+	96.6+	96.6+	96.8+	96.6+	96.5+	96.1
cmc	43.3	49.6+	48.3+	47.4+	45.0	46.1+	45.1+	46.2+
diabetes	70.8	74.6+	75.4+	75.8+	73.4	75.0+	73.8+	74.1+
ecoli	81.0	86.6+	86.6+	85.4+	85.1+	85.7+	82.7	83.3+
glass	65.9	64.5	65.9	64.0	61.2	63.1	65.4	65.0
heart	75.6	79.3	78.9	80.0	79.3	79.3	77.0	77.4
hill-valley	59.4	55.3–	59.4	49.7–	51.0–	48.5–	58.3	59.5
ionosphere	86.0	84.9	83.5–	84.1	84.3	84.9	86.9	86.3
iris	95.3	95.3	95.3	96.0	96.0	96.0	95.3	95.3
liver	61.5	62.9	64.1	62.3	57.3	60.6	61.7	62.0
musk2	73.8	74.7+	74.0+	78.4+	82.2+	76.0+	75.1+	74.9+
segmentation	96.3	95.5–	95.5–	94.6–	94.8–	94.8–	96.0	96.1
sonar	61.1	57.7–	59.6	59.1	60.6	60.6	62.9	62.0
spectf	71.5	75.6	74.9	72.7	74.2	75.3	71.9	73.4
vehicle	68.2	70.3	71.0+	68.3	64.7–	67.4	66.6	68.2
vowel	91.9	78.8–	80.9–	70.83–	80.9–	79.6–	87.5–	87.1–
waveform	77.2	80.8+	80.7+	79.8+	81.2+	78.9+	78.4+	78.0+
wine	94.9	94.4	94.9	94.4	94.4	93.8	94.9	94.9
yeast	48.8	57.0+	56.5+	56.1+	56.4+	55.3+	50.9+	50.1
%+/–		60/35	65/20	65/35	55/45	55/45	65/25	70/15
%sig +/–		40/20	45/15	40/15	30/20	40/15	35/5	30/5
Wilcoxon			Y				Y	Y

a '–' next to the accuracy figure and significant increases highlighted with a '+'. The percentage of datasets showing an increase or decrease in accuracy (labelled *% +/–*) are included with the percentage of datasets with significant increases and decreases also listed (labelled *%sig +/–*).

The Freidman test which considers the average rank of each algorithm indicates that there are no significant differences across the accuracy achieved by the different algorithms, however, the Wilcoxon Signed Ranks test which takes into account the magnitude of the differences by allowing greater differences to count more, shows that LSVM-rbf, BBNR and removing instances with a DL profile result in a higher generalisation accuracy than the unedited dataset where the difference is significant at the 5% level.

Looking at the individual results, no particular algorithm results in an improvement for each and every dataset. However, removing the DL instances performs consistently better than other algorithms. It only results in a reduction in generalisation accuracy for three of the 20 datasets (in two of these datasets (glass and segmentation) the reduction is less than 1 percentage point). For the remaining dataset (vowel) however, there is a significant dis-improvement – although none of the noise reduction algorithms result in improved accuracy on this dataset and the drop in generalisation accuracy for DL is considerably less (approx 5 point) compared with the other algorithms, in particular TER-S (a 22 point drop) or LSVM-lin (a 13 point drop). In fact, vowel appears to be a well-separated dataset with little noise which does not necessarily require editing.

TER and RENN appear to perform quite poorly, with the generalisation accuracy lower than that achieved on the unedited set almost half of the time. Although these algorithms can perform very well on certain datasets (e.g. almost 8 point and 7 point improvements in generalisation accuracy respectively on the yeast dataset) they also perform very badly on others including a decrease in generalisation accuracy of 10 and 11 points respectively on vowel and 9 and 11 points respectively on hill-valley. The performance of the more successful algorithms is more consistent and less variable across the datasets.

Although we are focusing here on competence enhancement, from the computational viewpoint TER is the slowest algorithm due to its internal leave-one-out procedure as discussed in Section 2.3. All of the other algorithms tested here are much faster than TER and are adequate at least for non-large datasets; however, due to their implementation on very different frameworks, their different approaches for model selection and the use of

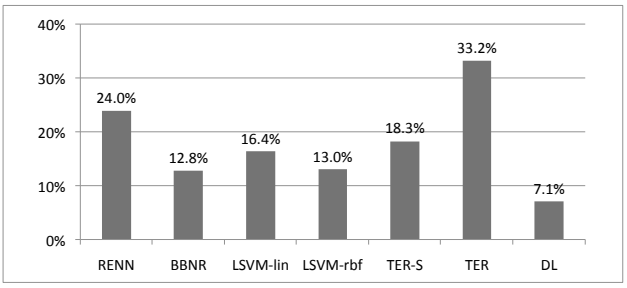


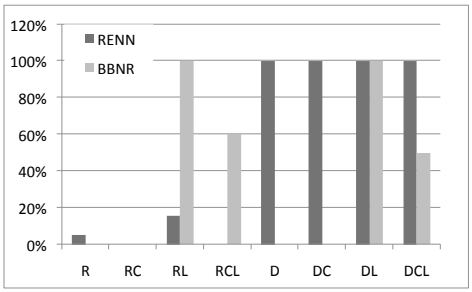
Figure 5: The percentage of instances removed by the different noise reduction algorithms averaged across all datasets.

different machines it is not worth evaluating their speeds.

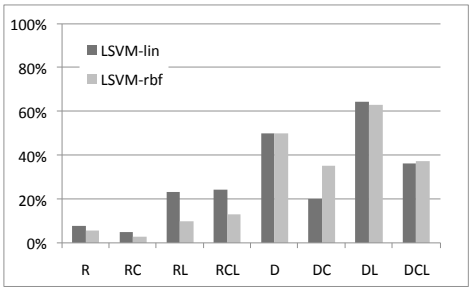
The proportion of the datasets removed by the different algorithms varies considerably. Figure 5 shows the overall percentage of instances removed by each noise reduction algorithm averaged (by median) across all datasets. All averages reported in this work are median averages to remove the effect of outliers as the distributions are quite skewed. RENN and the TER algorithms remove the highest number of instances, with TER on average removing one third of the dataset. This would seem to be high for an algorithm that claims to remove noise and in Section 4.1 we show how these algorithms do in fact focus on instances that can be considered redundant rather than noisy within the context of the training set. The techniques that focus on removing the smaller amounts of instances are those that perform best - including LSVM-rbf, BBNR and the approach of just removing damaging DL instances (with the latter reducing the datasets by the least amount, on average just over 7%).

4.1. Profiles of Instances Deleted by Noise Reduction Algorithms

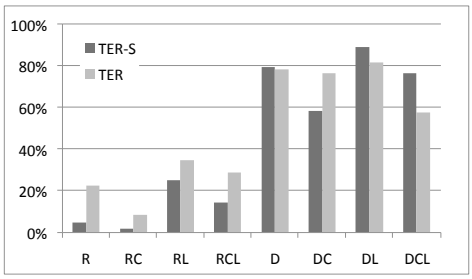
The profiling approach described in Section 3 offers the opportunity to consider the types of instances focussed on for removal by the different noise reduction techniques. Figure 6 shows the percentage of each type of instance in the original dataset that are removed by different algorithms averaged over all datasets whereas Figure 7 shows the proportion of the instances that are deleted which are of each type. To clarify the differences between these graphs, Figure 7(a) shows that on average 57% of the instances deleted by BBNR have a DL profile whereas Figure 6(a) shows that on average 100% of the DL instances in a dataset are removed by BBNR.



(a) RENN vs BBNR



(b) LSVM-lin vs LSVM-rbf



(c) TER vs TER-S

Figure 6: Percentage of each type of instance removed by the different noise reduction algorithms.

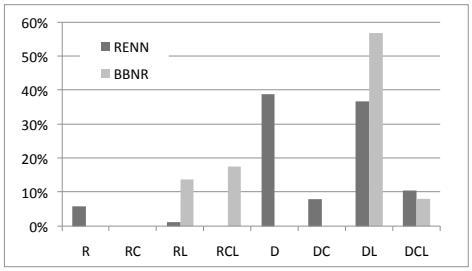
The most obvious focus of difference is between the RENN approach and the BBNR approach. RENN removes instances that are misclassified by the training set, so removes instances that have a D in their profile; whereas BBNR focusses on instances that are harmful in the training set causing other instances to be misclassified, indicated by having an L in their profile. Figure 7(a) illustrates this, with the majority of the deleted instances having D in their profile for RENN and all instances removed by BBNR having L in their profile. Figure 7(a) also shows that a small proportion of the instances removed by RENN include R in their profile which is contrary to what is expected. This is due to the fact that the algorithm is iterative; repeating until all instances in the resultant edited training set are classified correctly by the edited set. During these iterations the profiles of particular instances might change as other instances are removed from the dataset. However, as the RENN algorithm runs independently of our profiler we report on the initial profile of the dataset generated at the beginning of the RENN process.

We have shown that removing DL instances improves the generalisation accuracy of a training set. Both RENN and BBNR remove all these instances but also remove a considerable amount of other types of instances (see Figure 6(a)). In particular in addition to the DL instances RENN removes all instances with D in their profile (i.e. D, DC and DCL), which our investigations suggested did not need to be removed. RENN also removes a proportion of the instances with a profile of R which might seem to be redundant instances but our earlier investigation showed generalisation accuracy dis-improvements on their removal.

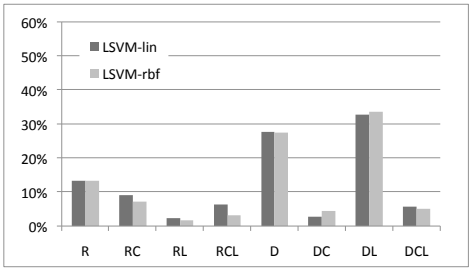
Figure 7(b) shows that there is little difference in the type of instances removed by the LSVM algorithms. Both algorithms perform well with LSVM-rbf performing the best. It's hard to get any significant evidence from the profiles of the instances removed to account for the difference. It seems that LSVM-lin focusses on removing marginally more of the beneficial R and RC instances than LSVM-rbf as can be seen in Figure 6(b). Also, LSVM-lin removes more instances on average than LSVM-rbf which may be a contributing factor. These are noise reduction algorithms that perform well in general and it may be possible to improve on their performance by adapting them to allow the RC and R instances to be kept by the editing process.

There are some differences between the performance of the two TER algorithms. Of the two, TER performs worst which is convenient as the varying threshold makes the algorithm unusable for large datasets. In addition TER is a very aggressive algorithm removing on average almost double the instances than its TER-S counterpart. It also focusses more on removing the beneficial R and RC instances, and removes less of the damaging DL instances as can be seen in Figure 6(c). Overall, TER is a noise reduction algorithm that does not seem to work well. Its simpler version TER-S does perform better, but its behaviour with respect to the type of instances it deletes is very similar to RENN, see Figures 8(a) and 8(b) and its performance in terms of generalisation accuracy is also very similar, as evident in Table 3.

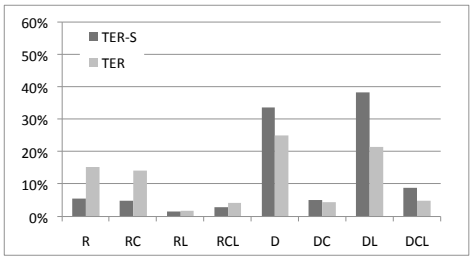
Another recent algorithm for instance selection is the Hit Miss Network (HMN) proposed by Marchiori [10] which focusses on competence preservation (redundancy removal) rather than noise reduction. Although variations on the original algorithm (HMN-E and HMN-EI) are adaptations for noise reduction, as the base algorithm focusses on redundancy reduction it was not included explicitly in the evaluation. However, this algorithm is interesting from the point of view of the RDCL profiles as it is straightforward to see the



(a) RENN vs BBNR

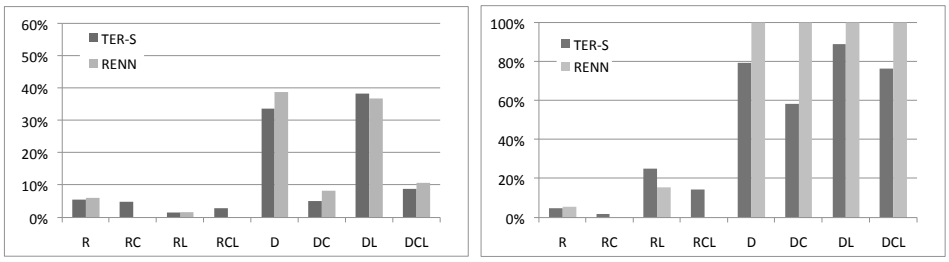


(b) LSVM-lin vs LSVM-rbf



(c) TER vs TER-S

Figure 7: The percentage of those instances removed by the different noise reduction algorithms which are of each type.



(a) % of deleted instances of each type (b) % of each type of instance deleted

Figure 8: Comparison of the behaviour of the TER-S and the RENN algorithms.

types of instances that it focusses on for removal. The algorithm creates a proximity directed graph from the nearest neighbour of each instance in the training set, with an edge connecting each instance to its nearest neighbour of each class. A 'hit' reflects the connection to the nearest like neighbour whereas a 'miss' reflects a connection to the nearest neighbour of a different class. The basic form of the algorithm removes the nodes in the graph with in-degree of zero which, in effect, removes instances with no coverage or liability sets, removing only instances with an R or D profile. Our evaluation in Section 3.2 showed that it was detrimental to overall generalisation accuracy to remove the R instances. Marchiori's results support this in that the basic form of their algorithm does not improve generalisation accuracy over a 1-NN classifier (see Table 2 in [10]).

4.2. Visualisation of Instances Deleted by Noise Reduction Algorithms

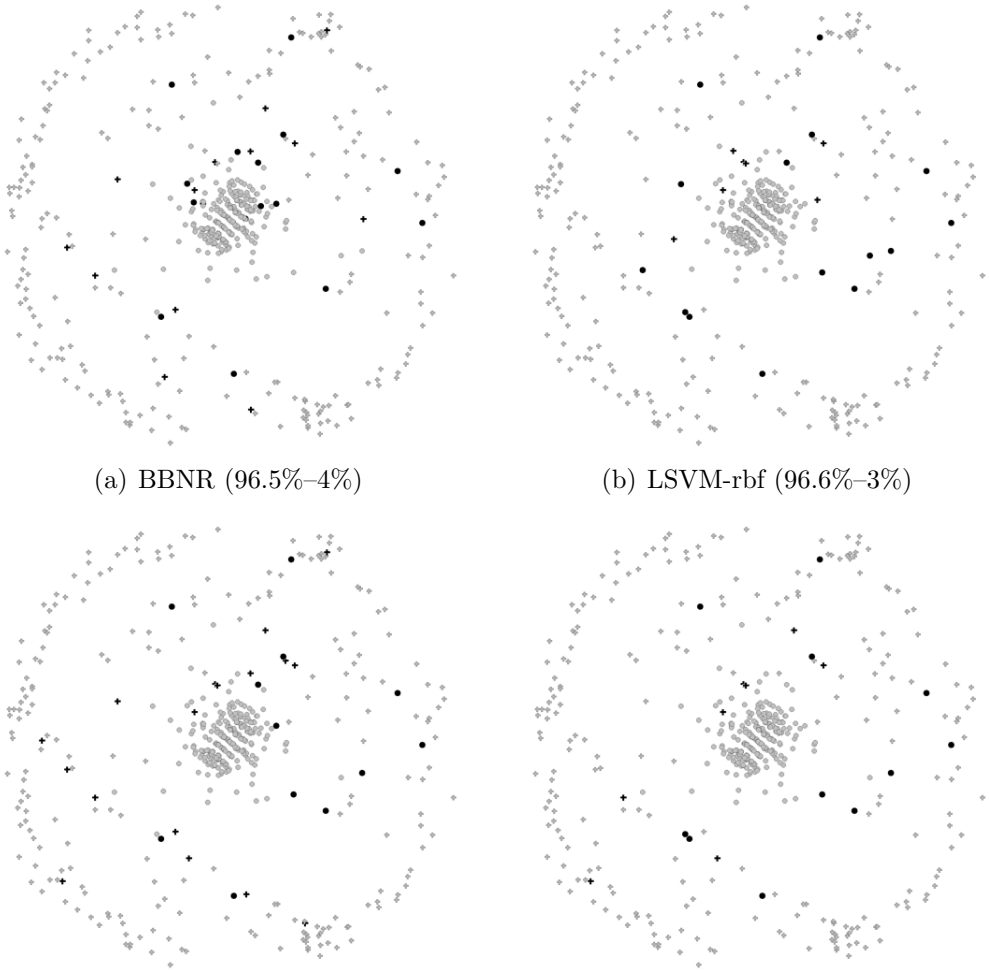
Figures 9 and 10 show visualisations of the breastcancer and liver datasets respectively, highlighting in black the instances which are removed by each of four different noise reduction techniques. The breast cancer dataset is an example of a well separated dataset where the cross validation accuracy on the original dataset is high and few instances are removed by any of the noise reduction techniques (less than 5% in all cases). It can be seen from Figure 9 that there is little difference across the different techniques in the instances removed, although by comparing with the visualization in Figure 3(c) interestingly, the LSVM-rbf techniques seems to keep more of the instances that could be considered to be more 'damaging', the DL instances,

Liver is an example of a less well separated dataset where the noise reduction techniques behave differently with regard to the instances that they remove. Figures 10(c) and 10(d) show little differences in the instances removed which reflects the conclusions above that there are a number of similarities between the RENN and TER algorithms. Both the LSVM-rbf and BBNR algorithms remove significantly less instances and Figures 10(a) and 10(b) show that they don't concentrate as much as the RENN and TER algorithms on removing large clusters of instances, in fact they each tend to remove different instances from the other. It is worth noting here too that by comparison with Figure 4(c) we can see that the LSVM algorithm leaves a number of the 'damaging' instances. Given that LSVM-rbf has the best performance on this dataset, this suggests that the local characteristics of this algorithm cause it to behave quite differently from the other algorithms in this study.

5. Conclusions

This paper has compared a number of noise reduction techniques that are currently in use in instance-based learning literature namely RENN, BBNR, LSVM-NR and TER. As the work is focussed on instance-based learning, our conclusions are less relevant for model based learners. We compared these noise reduction techniques not just from the perspective of comparative performance but from the perspective of identifying the types of instances that each focusses on for removal. We use the RCDL profiling technique which categorises each instance in a training set into one of eight categories based on its local competence properties. These categories capture whether the instance is correctly classified by the rest of the training set and whether it is beneficial and/or harmful depending on how it contributes to the classification of other instances in the training set.

It is obvious that instances in a training set that are classified correctly and contribute to the successful classification of others are beneficial to the training set. A significant finding of this paper is that we are able to identify such instances with the RCDL profiling technique and we have shown that removing such instances from a training set is significantly detrimental to the competence of the training set. We also found that the removal of instances that may appear to be redundant (i.e. not used in the classification of other instances in the training set) is not necessarily beneficial to generalisation accuracy.



(a) BBNR (96.5%–4%)

(b) LSVM-rbf (96.6%–3%)

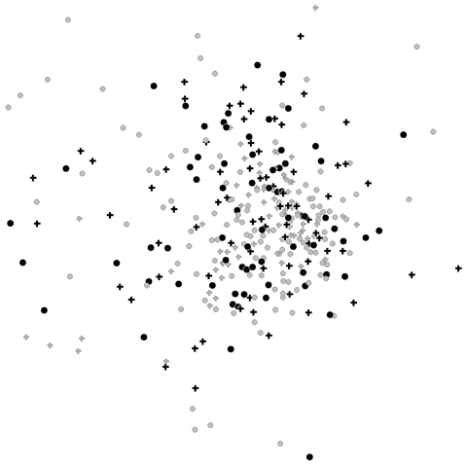
(c) RENN (96.6%–4%)

(d) TER-S (96.6%–3%)

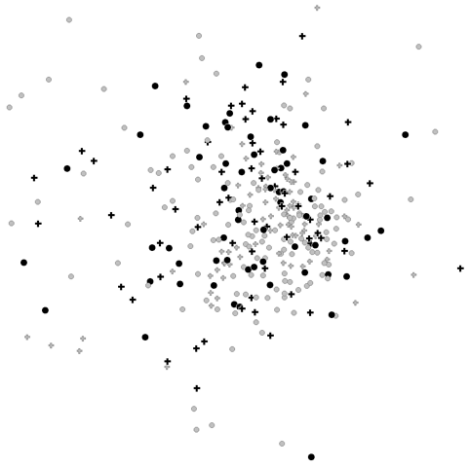
Figure 9: Visualisations showing the instances removed in the breast cancer dataset (highlighted in black) by each of the noise reduction techniques. The cross validation accuracy achieved and the percentage of instances removed by each technique is also shown (accuracy%–deleted%).

(a) BBNR (61.7%–24%)

(b) LSVM-rbf (64.1%–31%)



(c) RENN (60.6%–42%)



(d) TER-S (62.3%–36%)

Figure 10: Visualisations showing the instances removed in the liver dataset (highlighted in black) by each of the noise reduction techniques. The cross validation accuracy achieved and the percentage of instances removed by each technique is also shown (acc%–del%).

Comparing the actual performance of the algorithms we evaluated we have found that no single noise reduction algorithm can improve the generalisation accuracy consistently on all the twenty datasets used in our evaluation. However the simple approach of removing all instances which are misclassified by the training set and cause other instances to be misclassified works very well as an effective noise reduction technique. Its performance, along with the performance of BBNR and LSVM (using an rbf kernel), is significantly better than using the original dataset, i.e. not removing noise, across the datasets used in our evaluation. Our results also suggest that algorithms which are conservative in the proportion of instances removed perform considerably better than aggressive ones.

Considering the types of instances removed by the different algorithms, there is little overlap in RENN and BBNR whereas TER-S and RENN seem to target the same types of instances and perform very similarly–both quite poorly overall.

The performance of the LSVM-NR algorithm is good, particularly with an rbf kernel. We have shown that this algorithm does remove a percentage of beneficial instances, those with an R or RC profile. Future work could consider adapting the algorithm to retain these types of instances.

Apart from noise reduction, the profiling technique we have proposed here enables other investigations. Future work will include the application of the RCDL profiling technique to the redundancy reduction task in order to improve it. Also, the recently increasing study of active learning in which the learning procedure has the possibly of choosing unlabeled examples to be labeled, can take advantage from our work. For example, an active learning algorithm should not be interested in selecting unlabeled points that, once labeled, end up with a R or D profile, because instances with these profiles are not used for classifying other instances. Addressing the potential learning advantage in selecting an instance for labelling with respect to the profiles it can have, has the potential of improving active learning approaches.

References

[1] D. W. Aha, D. F. Kibler, Noise-tolerant instance-based learning algorithms, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89), Morgan Kaufmann Publishers Inc., 1989, pp. 794–799.

[2] G. W. Gates, The reduced nearest neighbor rule, IEEE Transactions on Information Theory 18 (1972) 431–433.

[3] P. E. Hart, The condensed nearest neighbor rule, IEEE Transactions on Information Theory 14 (1968) 515–516.

[4] G. L. Ritter, H. B. Woodruff, S. R. Lowry, T. L. Isenhour, An algorithm for a selective nearest neighbor decision rule, IEEE Transactions on Information Theory 21 (1975) 665–669.

[5] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms., Data Mining and Knowledge Discovery 6 (2002) 153–172.

[6] B. Smyth, M. Keane, Remembering to forget: A competence preserving case deletion policy for CBR systems, in: C. Mellish (Ed.), Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, (IJCAI-95), Morgan Kaufmann Publishers Inc., 1995, pp. 337–382.

[7] D. Wilson, T. Martinez, Instance pruning techniques, in: Proceedings of the Fourteenth International Conference on Machine Learning (ICML 97), Morgan Kaufmann Publishers Inc., 1997, pp. 403–411.

[8] N. Segata, E. Blanzieri, S. J. Delany, P. Cunningham, Noise reduction for instance-based learning with a local maximal margin approach, Journal of Intelligent Information Systems 35 (2009) 301–331.

[9] S. Massie, S. Craw, N. Wiratunga, When similar problems don't have similar solutions, in: Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR 07), Springer-Verlag, Berlin, Heidelberg, 2007, pp. 92–106.

[10] E. Marchiori, Hit miss networks with applications to instance selection, Journal of Machine Learning Research 9 (2008) 997–1017.

[11] D. L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man and Cybernetics 2 (1972) 408–421.

[12] I. Tomek, An experiment with the nearest neighbor rule, IEEE Transactions on Information Theory 6 (1976) 448–452.

[13] J. S. Sánchez, R. Barandela, A. I. Marqués, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (2003) 1015–1022.

[14] E. McKenna, B. Smyth, Competence-guided editing methods for lazy learning, in: W. Horn (Ed.), Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000), IOS Press, 2000, pp. 60–64.

[15] S. J. Delany, P. Cunningham, An analysis of case-based editing in a spam filtering system, in: P. Funk, P. González-Calero (Eds.), Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR 2004), volume 3155 of *LNAI*, Springer, 2004, pp. 128–141.

[16] S. J. Delany, The good, the bad and the incorrectly classified: Profiling cases for case-base editing, in: L. McGinty, D. C. Wilson (Eds.), Proceedings of the Eighth International Conference on Case-Based Reasoning (ICCBR 09), volume 5650 of *LNCS*, Springer, 2009, pp. 135–149.

[17] P. Devijver, J. Kittler, Pattern recognition: a statistical approach, Englewood Cliffs, London (1982).

[18] D. R. Wilson, T. R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (2000) 257–286.

[19] B. B. Chaudhuri, A new definition of neighborhood of a point in multi-dimensional space, Pattern Recognition Letters 17 (1996) 11 – 17.

[20] A. Vezhnevets, O. Barinova, Avoiding boosting overfitting by removing confusing samples, in: J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, A. Skowron (Eds.), Proceedings of the 18th European Conference on Machine Learning (ECML), volume 4701 of *LNCS*, Springer, 2007, pp. 430–441.

[21] M. Salamó, M. López-Sánchez, Adaptive case-based reasoning using retention and forgetting strategies, Knowledge-Based Systems 24 (2011) 230 – 247.

[22] G. Singh, Improving CBR: Investigating improved approaches to Case Base Maintenance and Textual Case Retrieval, PgDip(Res), School of Computing, Dublin Institute of Technology, 2010.

[23] E. Blanzieri, F. Melgani, An adaptive SVM nearest neighbor classifier for remotely sensed imagery, in: Proceedings of the IEEE International Conference on Geoscience and Remote Sensing Symposium (IGARSS 2006), pp. 3931–3934.

[24] E. Blanzieri, F. Melgani, Nearest neighbor classification of remote sensing images with the maximal margin principle, IEEE Transactions on Geoscience and Remote Sensing 46 (2008) 1804–1811.

[25] N. Segata, E. Blanzieri, Fast and scalable local kernel machines, Journal of Machine Learning Research 11 (2010) 1883–1926.

[26] N. Segata, E. Blanziere, P. Cunningham, A Scalable Noise Reduction Technique for Large Case-based Systems, in: 8th International Conference on Case-based Reasoning (ICCBR 09), volume 5650 of *LNCS*, Springer, 2009, pp. 328–342.

[27] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, in: Proceedings of the 23rd International Conference on Machine Learning (ICML '06), ACM, New York, NY, USA, 2006, pp. 97–104.

[28] P. Eades, A Heuristic for Graph Drawing, Congressus Numerantium 42 (1984) 149–160.

[29] B. Smyth, M. Mullins, E. McKenna, Picture perfect: Visualisation techniques for case-based reasoning, in: W. Horn (Ed.), Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000), IOS Press, 2000, pp. 65–72.

[30] B. Mac Namee, S. J. Delany, CBTV: Visualising case bases for similarity measure design and selection, in: I. Bichindaritz, S. Montani (Eds.), Case-Based Reasoning. Research and Development, 18th International Conference on Case-Based Reasoning, ICCBR 2010, Proceedings, volume 6176 of *LNCS*, Springer, 2010, pp. 213–227.

[31] B. Mac Namee, R. Hu, S. J. Delany, Inside the selection box: Visualising active learning selection strategies, in: Proceedings of the Challenges of

Data Visualization Workshop at the Twenty-Fourth Annual Conference on Neural Information Processing Systems.

[32] A. Asuncion, D. Newman, UCI machine learning repository, 2007. University of California, Irvine, School of Information and Computer Sciences.

[33] Q. McNemar, Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika 12 (1947) 153–157.

[34] J. Demsar, Statistical comparison of classifiers over multiple data sets, J. Machine Learning Research 7 (2006) 1–30.

[35] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80 – 83.

[36] M. Freidman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (1937) 675–701.

[37] M. Freidman, A comparison of alternative test of significance for the problem of m rankings, Annals of Mathematical Statistics 11 (1940) 86–92.

[38] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.