Data Mining and Knowledge Discovery, 6, 153–172, 2002

Advances in Instance Selection for Instance-Based Learning Algorithms*

HENRY BRIGHTON henryb@ling.ed.ac.uk
Language Evolution and Computation Research Unit, Department of Theoretical and Applied Linguistics, The University of Edinburgh, Edinburgh, EH8 9LL, UK

CHRIS MELLISH chrism@dai.ed.ac.uk
Department of Artificial Intelligence, The University of Edinburgh, Edinburgh EH1 1HN, UK

Editors: Mannila, Liu, Motoda

Received September 18, 1999; Revised January 15, 2001

Abstract. The basic nearest neighbour classifier suffers from the indiscriminate storage of all presented training instances. With a large database of instances classification response time can be slow. When noisy instances are present classification accuracy can suffer. Drawing on the large body of relevant work carried out in the past 30 years, we review the principle approaches to solving these problems. By deleting instances, both problems can be alleviated, but the criterion used is typically assumed to be all encompassing and effective over many domains. We argue against this position and introduce an algorithm that rivals the most successful existing algorithm. When evaluated on 30 different problems, neither algorithm consistently outperforms the other: consistency is very hard. To achieve the best results, we need to develop mechanisms that provide insights into the structure of class definitions. We discuss the possibility of these mechanisms and propose some initial measures that could be useful for the data miner.

Keywords: instance-based learning, instance selection, forgetting, pruning

1. Introduction

The Nearest Neighbour Classifier is a simple supervised concept learning scheme which classifies unseen (i.e., unclassified) instances by finding the closest previously observed instance, taking note of its class, and predicting this class for the unseen instance (Cover and Hart, 1967). Learners that employ this classification scheme are also termed Instance-Based Learners, Lazy Learners, Memory-Based Learners, and Case-Based Learners. They all suffer from the same problem: the instances used to train the classifier are stored indiscriminately. No process of selection is performed, and as result, harmful and superfluous instances are stored needlessly. Disregarding this problem, the classification scheme is simple and very effective in comparison to other methods such as feed-forward neural networks and decision trees (King et al., 1995).

*This work was partially supported by the UK ESRC, and Sharp Laboratories of Europe Ltd, Oxford.

In this article we survey the chief efforts to alleviate this problem and review the criteria used to selectively store instances of the classification problem. We review this work using insights about the structure of classification problems in general. By viewing instances as feature vectors we can imagine an instance space where each instance is a point. We argue that the structure of the classes formed by the instances can be very different from problem to problem, which results in inconsistency when we apply one instance selection scheme over many problems. The thrust of this article is that the data miner needs to gain an insight into the structure of the classes within the instance space to effectively deploy an instance selection scheme. We shed light on possible class structures, and how they can be grouped. We aim to show that a knowledge of the class structures is an intrinsic part of designing and deploying instance selection algorithms.

The structure of this article is as follows. In Section 2 we characterise the problem by discussing exactly what an instance selection algorithm should achieve, and in what circumstances this is possible. We argue that different kinds of problem spaces, specifically the class structures, require a different interpretation of what a critical instance is. Using these insights we review previous work in Section 3. We group previous work on the basis of what aspect of the problem they attempt to solve: noise removal, competence preservation, and those that attempt both objectives. We then address the problem of how to compare these algorithms: little comparative work has been carried out in the past. We argue that three eras have occurred in the development of instance selection algorithms, with the most recent approaches being superior. Our contribution to the evaluation is a comparison of the ICF algorithm (Brighton and Mellish, 1999) with RT3 (Wilson and Martinez, 1997) over 30 domains. We argue that neither of these two algorithms is superior: both record the highest accuracy and space reduction on certain problems. In the context of instance-based learning, both algorithms represent the cutting edge in instance set reduction techniques. Finally we discuss how our ICF algorithm offers insights into the structure of the instance space, and we discuss some future research directions.

2. Defining the problem

We want to isolate the smallest set of instances which enable us to predict the class of a query instance with the same (or higher) accuracy than the original set. Before reviewing the many methods one can employ to tackle this problem, we present two practical issues which are often neglected. First, we point out that instance selection is practically realised by instance removal in the context of nearest neighbour classification: we aim to retain only the critical instances. We argue that any scheme should aim to achieve what we term unintrusive storage reduction, which defines the position we should aim for in the trade-off between storage reduction and classification accuracy. Secondly, we argue that in the context of instance selection, we need to differentiate between certain types of classification problem: domains with homogeneous class definitions and those without. We argue that different removal criteria are required for the two opposing class structures. The second point reinforces the thrust of this article: consistency over many problems is hard when designing an instance filter. Instead of placing the whole solution on the algorithm, we argue it is largely placed on the data miner, as a knowledge of problem structure is required to select the best tool. In Section 5 we propose some measures to aid this process.

2.1. Selection as removal: How to preserve classification competence

In general we define the problem of instance selection as the need to extract the most useful set of instances from a database which we know (or suspect) contains instances which are superfluous or harmful. In the context of instance-based learning, we seek to discard the cases which are superfluous or harmful to the classification process. Some instances of a class are just not telling us much, the job they do in informing classification decisions is done far better by other cases: they are superfluous. Similarly, some instances of a class might lead us to make false classification predictions if we rely on them: they are harmful.

In the context of instance-based learning, the problem of instance selection should be viewed more in terms of instance deletion as we remove superfluous and harmful instances and retain only the critical instances. By removing a set of instances from a database the response time for classification decisions will decrease, as fewer instances are examined when a query instance is presented. This objective is primary when we are working with large databases and have limited storage. The removal of instances can also lead to either an increase or decrease in classification competence. Therefore, when applying a instance selection scheme to a database of instances we must be clear about the degree to which we are willing to let the original classification accuracy depreciate. For example, if we have a fixed storage limit then the number of cases we are forced to remove might be too large, and unavoidably result in a degradation of classification accuracy. For example, the schemes used by Markovitch and Scott (1993) and Smyth and Keane (1995) employ fixed storage limits. Usually, the principle objective of an instance selection scheme is unintrusive storage reduction. Here, classification accuracy is primary: we desire the same (or higher) classification accuracy but we require it faster and taking up less space. Ideally, accuracy should not suffer at the expense of improved performance.

Now, if our deletion decisions are not to harm the classification accuracy of the learner, we must be clear about the kind of deletion decisions that introduce erroneous classification decisions. Consider the following reasons why a k-nearest neighbour classifier might incorrectly classify a query instance:

1. When noise is present in locality of the query instance. The noisy instance(s) win the majority vote, resulting in the incorrect class being predicted.
2. When the query instance occupies a position close to an inter-class border where discrimination is harder due to the presence of multiple classes.
3. When the region defining the class, or fragment of the class, is so small that instances belonging to the class that surrounds the fragment win the majority vote. This situation depends on the value of k being large.
4. When the problem is unsolvable by an instance-based learner. This will be due to the nature of the underlying function, or due to the sparse data problem.

In the context of instance selection, we can address point (1) and try and improve classification accuracy by removing noise. We can do nothing about (4) as this situation is a given and defines the intrinsic difficulty of the problem. However, issues (2) and (3) should guide our removal decisions. Removing instances that are close to borders is not recommended as these instances are relevant to discrimination between classes. We should be aware of point (3), but as k is typically small, the occurrence of such a problem is likely to be rare.

An interesting point to note here which, to our knowledge, has not been made before is that one can place a theoretical limit on how our instance-base reduction algorithms should perform. In practice, we take a small random sample of our classification problem and keep these instances for testing the accuracy of the nearest neighbour classifier. Now, given an instance-base I, we form two sets: *training* and *testing*. That is, $\textit{training} \cup \textit{testing} = I$. If we then make the following assumptions:

1. $|\textit{training}| > |\textit{testing}|$
2. We are using 1-nearest neighbour classification.

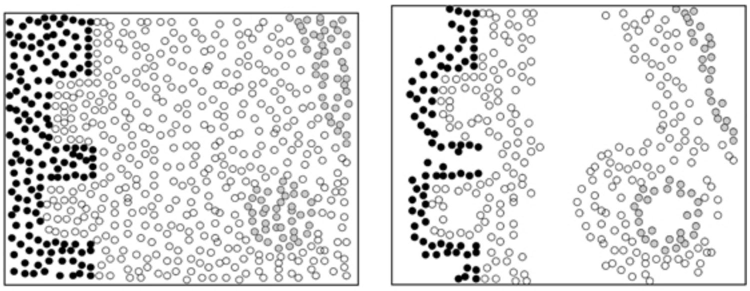
Then we can say that, after filtering *training*, the maximum number of instances in *training* required by the classifier to retain its original classification accuracy is in fact $|\textit{testing}|$. This result follows as for each instance in our testing set, from which we derive the accuracy of the classifier, we only need one case in *training* to correctly classify that one test instance. We can use this result as a guide to check if our algorithms are performing as they should. We say that we should be left with a maximum of $|\textit{testing}|$ cases after instance selection is complete. The minimum number of instances required to retain classification accuracy on the testing set gives us a measure of how easy the problem is. We use this observation later.

2.2. *The structure of the instance space*

Traditionally, the way in which critical instances are identified in an instance space is assumed to apply to all classification problems: we desire an algorithm which we can apply to any domain. We argue against this position and propose two broad categories of class structure which require dramatically different approaches.

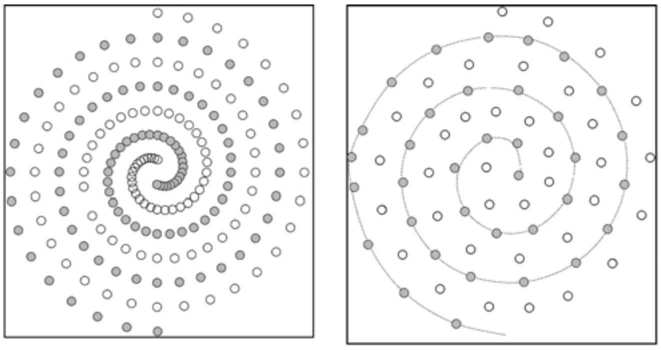
Fortunately, the vast majority of problems we encounter, especially in the field of data mining, fall into a single category. This category contains instance spaces whose classes are defined by homogeneous regions of instances. To illustrate such an instance space figure 1(a) depicts the 2d-dataset which we constructed to visualise instance selection decisions. The three classes (black, grey, and white) are each defined by regions of instances which share a class, i.e., each white instance is usually in the locality of other white instances. The second category is composed of those problems which have classes defined by non-homogeneous regions. For example, problems such as the two-spirals dataset depicted in figure 2(a). Here, the classes are represented by a spiral structure which is not localised to one region of the space. In the past, the characterisation of a critical instance has not been problem dependent, partly due to rarity of non-homogeneous class structures amongst machine learning and data mining problems.

Given a class defined by a homogeneous collection of instances, which instances are critical to classifying instances of that class? There are many approaches which tackle this question. For example, we might aim to identify instances that are prototypes (Chang,



(a) (b)

Figure 1. (a) The 2d-dataset, which is composed of homogeneous class definitions. (b) Removing instances from the interior of class definitions does not lead to a drop in classification accuracy as discrimination is still possible.



(a) (b)

Figure 2. (a) The two-spirals dataset, an example of a problem space not defined by homogeneous collections of cases. (b) Chang's prototype creation algorithm retains the class structure well.

1974; Zhang, 1992, Sebban et al., 1999) or instances with high utility (Markovitch and Scott, 1988, 1993; Smyth and Keane, 1995; Aha et al., 1991). We argue, as others have (Swonger, 1972; Wilson and Martinez, 1997), that instances which lie on borders between classes are almost always critical to the classification process. The instances located at the interior of class regions are superfluous as their removal does not lead to any loss in the ability of the nearest neighbour learner to discriminate between classes, which, for us, is the purpose of classification. To illustrate this point, the set of instances shown in figure 1(b) will correctly predict queries just as well as those instances in figure 1(a). A prototype, an instance which in some way represents the essence, average, or typicality of a class is useful in characterising a class, but not in characterising the differences between classes.

Instances with high utility may well turn out to be critical border cases, but this is in no way guaranteed, as the manner in which we identify these instances is not guided by our presupposition that border cases are critical. The problem with utility-based methods is that we require a knowledge of the prior use of instances, i.e., classification feedback.

When instances have not been used, they may have an inaccurate measure of utility. Indeed, border cases are less likely to be excel in this framework, as interior cases are by definition surrounded by cases of their own class and will therefore:

1. Have a high probability of predicting queries correctly.
2. Have a high probability of being used as a classifier.

We can define a non-homogeneous class as one which is defined by a group of instances not sharing the same locality. Here, the notion of a border instance doesn't make sense. One might argue that all of the instances make up the borders and they are therefore critical to the definition of the class; instance selection is a bad proposition when working with problems of this form. We argue that in this kind of situation keeping only prototypical instances is the safest way to remove a number of instances. For example, we can dramatically reduce the number of instances in the two-spirals dataset by a employing prototype selection algorithm. Figure 2(b) shows the remaining prototypes after applying Chang's algorithm, discussed later. The class structure is still well defined.

2.3. Summary

To summarise, we have shown that the nature of a critical case depends on the structure of the class definitions. The majority of problems we find fall into the first category: the classes are defined by homogeneous regions of instances. We must be aware of other types of class structure. Given this skew towards homogeneous class structures we argue that prototypes might be good classifiers because they can classify many instances in the instance space. However, they are not good discriminators.

3. Review

Selectively storing the set of presented instances has been an issue since the early work on nearest neighbour classification. The early schemes typically concentrate on either competence enhancement (noise removal) or competence preservation. We define these schemes as follows:

1. *Competence enhancement*: By removing certain instances it is often possible to increase the classification accuracy of the learner. This is possible when noisy or corrupt instances are isolated and removed.
2. *Competence preservation*: A superfluous instance is one which, when removed, will not lead to a decrease in classification accuracy. We can therefore remove it without any loss in classification competence.

In general, noise removal schemes will result in few cases being removed, and with little chance of competence depreciation and a high chance of competence enhancement. On the other hand, schemes which aim to preserve competence typically remove many cases, but are unlikely to result in competence enhancement. We will group previous studies on the basis of this distinction as well as introducing another distinction for those schemes that tackle both problems. We term the third group the hybrid approaches. Most modern instance selection algorithms are hybrid approaches.

We have chosen, for the sake of brevity, not to consider work which addresses the problem by choosing alternative, more efficient representations (for example, Salzberg (1991), Domingos (1995), Daelemans et al. (1997)). We view such work as a separate issue as individual instances are not identified or removed: the schemes compress rather than filter. Also, in the context of data mining one might be restricted, and have to employ a conventional database representation. All the schemes we discus here are applicable to a simple database model. Other interesting removal criteria have been proposed in the context of concept drift (Salganicoff, 1993). We will not investigate such issues, as we assume our class definitions are true over time. Few reviews have been compiled in this area, although a good collection of the early schemes, as well as a good overview is provided by Dasarathy (1991), but unfortunately, no experimental comparison is made between the methods.

3.1. Competence enhancement

Noise can occur for a number of reasons, and takes many forms. We restrict our treatment of noise to what we term pointwise miss-labellings. We assume no pattern in the noise other than a random peppering of miss-labelled instances.

The first scheme we discuss is Wilson Editing (Wilson, 1972), which attempts to remove noisy instances by making a pass through all the instances in the training set and removing those which satisfy an editing rule. The rule is simple: all instances which are incorrectly classified by their nearest neighbours are assumed to be noisy instances. The instances which satisfy such a rule will be those that have a different class to their neighbour(s). These instances will appear as exceptions within regions containing instances of the same class. Other candidates fulfilling this rule could be the odd instance lying on a border between two different classes. For this reason, Wilson Editing can be thought of as smoothing the instance-space at it removes instances that deviate from the coherent regions defined by instances sharing the same class. Wilson reported improved classification accuracy over a large class of problems when using the edited set rather than the original, unfiltered set. Tomek (1976) compared Wilson's algorithm with two new methods: Repeated Wilson Editing, and All k-NN. Repeated Wilson Editing is identical to Wilson's approach described above, only it is carried out repeatedly, until the rule is not applicable to any more cases. This approach can result in better noise detection than the basic algorithm when more than one occurrence of noise is present in a locality. The all-kNN algorithm is similar, only after each iteration the value k is increased. In the average case these algorithms result in improved classification competence, but storage reduction is not significant as only noisy and fringe instances are removed. Figure 3 illustrates the result of noise removal for a simple problem.

The problem with these schemes is that they will only work when a small amount of noise is present. If we introduce a high percentage of noise then the noisy instances will no longer appear as exceptions as these noisy instances will start being correctly classified by other noisy instances. An important point to note when removing harmful instances is that in certain domains, we cannot differentiate between noise and genuine class exceptions.

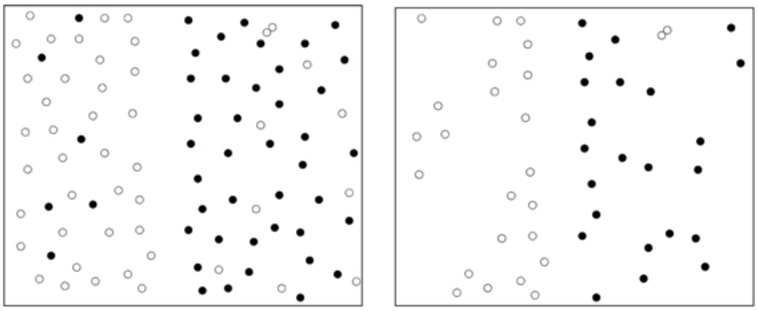


Figure 3. (a) A simple two class problem in which noisy instances have been introduced. (b) Repeated Wilson Editing removes all the noisy instances except the two instances of noise which lie next to each other.

Recent work by Daelemans et al. (1999) suggests that natural language domains, such as word pronunciation, are problematic in the context of instance deletion as the class definitions are not composed of large homogeneous regions but rather many small regions or exceptions (also termed small disjuncts by Holte et al. (1989)). Deleting an instance in this kind of situation is a real problem, and reinforces the point we make in Section 2: we need a knowledge of the problem to effectively deploy a deletion scheme.

3.2. *Competence preservation*

The majority of work carried out on competence preservation occurred shortly after the inception of nearest neighbour classification. One problem with work from this era is that many of the algorithms were not compared with each other, and when they were, the algorithms were only evaluated on a few classification problems.

Hart's (1968) Condensed Nearest Neighbour rule (CNN) was an early attempt at finding, using Hart's terminology, a minimally consistent subset of the training set. A consistent subset of a training set T is some subset S of T that correctly classifies every case in T with the same accuracy as T itself. A consistent subset is therefore likely to preserve the classification accuracy achieved on the testing set. The deletion criteria used by the CNN is the opposite of that used in Wilson editing. Instead of looking to label cases which are misclassified by T as noise, we are looking for cases for which removal does not lead to additional miss-classifications. This criterion therefore results in superfluous cases being weeded out. The CNN algorithm seeks a minimal consistent subset but is not guaranteed to find one.

Gates (1972) devised the Reduced Nearest Neighbour (RNN) rule, which extends the idea of the CNN by contracting a complete set of instances to form a consistent subset. Again, the ideal is to find a minimally consistent subset. Both Hart and Gates use the same criteria for case deletion, but build the edited set of cases from opposite starting positions. The criterion used is essentially that of learning feedback: for a case to be kept it must prove useful on the basis of classification trials. Both schemes are highly likely to retain noise.

The Selective Nearest Neighbour Rule (SNN) devised by Ritter et al. (1975) improves on the CNN and RNN by ensuring that a minimal consistent subset is found. The selection criteria is strict by enforcing the following rule: all instances in the training set must be closer to an instance in the selective set than any instance of a different class found in the training set. Ritter et al. (1975) reported improved prediction accuracy when compared to the CNN.

Chang's algorithm (1974) offers a novel approach to removing cases by repeatedly attempting to merge two existing cases into a new case. The process of merging cases results in a case-base containing cases which were not actually observed, but rather constructed. These cases are termed prototypes, which we can view as synthetic cases derived from the exemplars which a traditional nearest neighbour classification scheme would use. Chang's algorithm searches for candidates for merging: We seek two cases p and q which we can replace with a single case z. The merging process is permitted when p and q are of the same class, and after replacing them with z, the consistency of the case-base is not breached.

Another novel approach to competence preservation is the Footprint Deletion policy of Smyth and Keane (1995) which is a filtering scheme designed for use within the paradigm of Case-Based Reasoning (CBR). We discuss this work here as Footprint Deletion provides a novel approach to the problem of case deletion which is relevant to our discussion. In previous work (Brighton, 1997) we have shown that some of the concepts introduced by Smyth and Keane transfer to the simpler context of the nearest neighbour classification algorithm. CBR is an approach to solving reasoning and planning tasks on the basis of past solutions (Kolodner, 1993). The technicalities are much the same as instance-based learning, although the concept of case adaptation is usually used as a similarity metric. A CBR system aims to solve a new task by adapting a previously stored solution in such a way that it can be applied to the new problem. Much of Smyth and Keane's work relies on the notion of case adaptation. They use the property $Adaptable(c, c')$ to mean case c can be adapted to c'. Generally speaking, we can delete a case for which there are many other cases that can be adapted to it. In our previous work we introduced a nearest neighbour parallel termed the Local-Set of a case c to capture this property (Brighton, 1996) (see figure 4). We define the Local-set of a case c as:

> The set of cases contained in the largest hypersphere centred on c such that only cases in the same class as c are contained in the hypersphere.

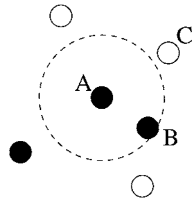


Figure 4. $LocalSet(A) = \{A, B\}$ as instance C bounds the hypersphere extended from A.

The originality of Smyth and Keane's work stems from their proposed taxonomy of case groups. By defining four case categories, which reflect the contribution to overall competence the case provides, we gain an insight into the effect of removing a case. We define these categories in terms of two properties: *Reachability* and *Coverage*. These properties are important, as the relationship between them has been used in crucial work which we discuss later. For a case-base $\mathcal{CB} = \{c_1, c_2, \ldots, c_n\}$, we define *Coverage* and *Reachability* as follows:

$$Coverage(c) = \{c' \in \mathcal{CB} : \text{Adaptable}(c, c')\} \tag{1}$$

$$Reachable(c) = \{c' \in \mathcal{CB} : \text{Adaptable}(c', c)\} \tag{2}$$

Using these two properties we can define the four groups in the taxonomy using set theory. For example, a case in the pivotal group is defined as a case with an empty reachable set. For a more thorough definition we refer the reader to the original article. Our investigation into the instance-based learning parallel of Footprint Deletion differs only in the replacement of Adaptable with the Local-set property. Whether a case c can be adapted to a case c' relies on whether c is relevant to the solution of c'. In lazy learning this means that c is a member of nearest neighbours of c'. However, we cannot assume that a case of a differing class is relevant to the solution (correct prediction) of c'. We therefore bound the neighbourhood of c' by the first case of a differing class. Armed with this parallel we found that Footprint deletion performed well. Perhaps more interestingly, we found that a simpler method which uses only the local-set property, and not the case taxonomies, performs just as well. With local-set deletion, we choose to delete cases with large local-sets, as these are cases located at the interior of class regions. The issue of deciding how many cases to delete is the problem. We chose to use Smyth and Keane's methodology of imposing a swamping limit, which is a pre-defined storage constraint. This contrasts with other algorithms, which typically decide dynamically when to stop removing cases. Local-set deletion has subsequently been employed in the context of natural language processing (van den Bosch and Daelemans, 1998).

3.3. Hybrid approaches

Aha et al. (1991) introduced the incremental lazy learning algorithms IB1, IB2, IB3, and IB4. We concentrate on IB2 and IB3 as their primary function is to filter training cases. With IB2, if a new case to be added can already be classified correctly on the basis of the current case-base, then the case is discarded and not stored at all. Only those cases which the learner can not classify correctly are stored. This is a measure employed to weed out superfluous cases, and is a good one as the cases never need to be stored, unlike the other algorithms reviewed here which operate on a batch of cases. The problem with IB2 is its susceptibility to harmful cases. Harmful cases will nearly always be stored, as they are exceptions and will therefore generally be misclassified. IB3 addresses this problem. IB3 is IB2 augmented with a "wait and see" policy for removing noisy cases. IB3 does this by keeping a record of how well the stored cases are classifying. Noisy cases are likely to be bad classifiers, so we can try and spot them after their inclusion in the case-base. Stored cases that miss-classify to a statistically significant degree are removed. Note that these cases could also be useful exceptions to the class definitions. A number of workers have augmented the IBn algorithms (Cameron-Jones, 1992; Zhang, 1992; Brodley, 1993). To summarise, Aha's algorithms offer an incremental approach to filtering, and for this reason offer improved efficiency, but suffer from the order of case presentation. Crucial cases could be rejected early on when the class definitions are poorly defined.

Wilson and Martinez (1997) present three algorithms for reducing the size of case-bases: RT1, RT2 and RT3. RT1 is the basic removal scheme. The algorithm proceeds by computing, for each case, the set of k nearest neighbours (where k is small and odd). Then, another set of cases is computed for each case p, termed the associates of the case p. The associates of case p are the set of stored cases which have p as one of their nearest neighbours. The set of nearest neighbours is always of size k, whereas the size of the set of associates can be larger. RT1 removes a case p if at least as many of its associates, after the removal of p, would be classified correctly without it, i.e., we look to see if removing a case p has a detrimental effect on those cases which have p as a nearest neighbour.

The removal of noise is implicit in this scheme. Noise will typically not lead to an increase in misclassification of its surrounding neighbours. It will therefore often be deleted by RT1. RT2 is identical to RT1, only the cases in the training set are sorted by the distance from their nearest enemy (a case of another class). Cases furthest from a case of another class are therefore deleted first. This means that cases furthest from boundary positions will be removed before cases in border areas. RT2 also differs from RT1 in that deletion decisions still rely on the original set of associates. A case can therefore have associates which have already been deleted, but are still used to guide case deletion as we continue to test the ability to classify them. RT3 differs from RT2 though the introduction of a noise filtering pass being executed before the RT2 procedure is carried out. The noise filtering procedure is similar to that of Wilson's (1972): remove those cases which are misclassified by their k nearest neighbours.

The RT algorithms are driven by the relationship between the nearest neighbours and the associates of each case. The relationship is analogous to that introduced by Smyth and Keane, where Coverage and Reachability are defined in terms of the *Adaptable* property. The properties used in the RT algorithm, those of bounded neighbourhood and associate sets, are similar to the relationships we used in implementing Smyth and Keane's work in the context of lazy learning. The algorithms differ in how they use these relationships, however. Wilson and Martinez have shown RT3 to consistently be the best case filter in a comparison with IB3.

3.3.1. An iterative case filtering algorithm. We now introduce our Iterative Case Filtering Algorithm (ICF) (Brighton and Mellish, 1999). The ICF algorithm uses the lazy learning parallels of case coverage and reachability we developed when transferring the CBR footprint deletion policy, discussed above. Rather like the Repeated Wilson Algorithm investigated by Tomek, we apply a rule which identifies cases that should be deleted. These cases are then removed, and the rule is applied again, iteratively, until no more cases fulfil the pre-conditions of the rule.

The ICF algorithm uses the reachable and coverage sets described above, which we can liken to the neighbourhood and associate sets used by Wilson and Martinez. An important difference is that the reachable set is not fixed in size but rather bounded by the nearest case of different class. This difference is crucial as our algorithm relies on the relative sizes of these sets. Our deletion rule is simple: we remove cases which have a reachable set size greater than the coverage set size. A more intuitive reading of this rule is that a case *c* is removed when more cases can solve *c* than *c* can solve itself. These cases will be those furthest from the class borders as their reachable sets will be large. After removing these cases the case-space will typically contain thick bands of cases either side of class borders.

This is the deletion criterion the algorithm uses; the algorithm proceeds by repeatedly computing these properties after filtering has occurred. Usually, additional cases will begin to fulfil the criteria as thinning proceeds and the bands surrounding the class boundaries narrow. After a few iterations of removing cases and re-computing, the criterion no longer holds. This point turns out to be a very good point to stop removing cases as removing more cases tends to breach our objective of unintrusive storage reduction. Figure 5(a)–(d) illustrates how the algorithm progresses.

The algorithm is depicted in figure 6. As with the majority of algorithms that concentrate on removing superfluous cases, ours is likely to protect noisy cases. A noisy case will have a singleton reachable set and a singleton coverage set. This property protects the case from removal. For this reason we employ the noise filtering scheme based on Wilson Editing and adopted by Wilson and Martinez. Lines 2–6 of the algorithm perform this task.

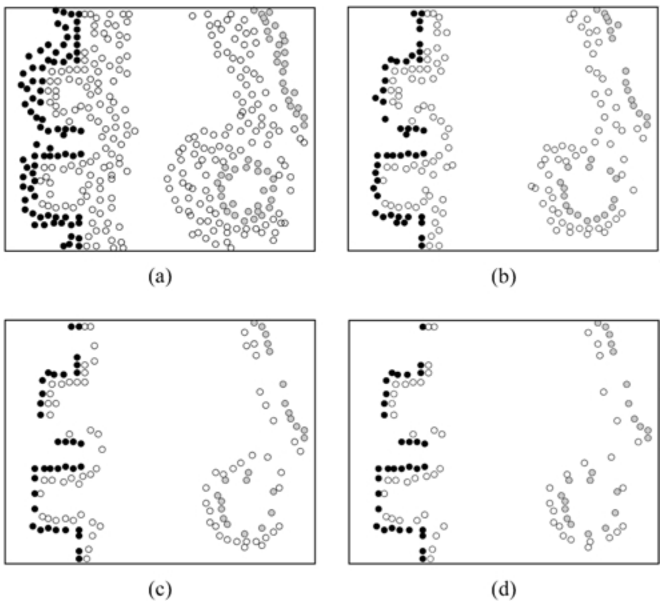


Figure 5. (a) The cases remaining from the 2d-dataset after 1 iteration of the ICF algorithm. (b) after 2 iterations, (c) after 3 iterations, and (d) after 4 iterations.

```
ICF(T)
> Perform Wilson Editing
for all x ∈ T do
    if x classified incorrectly by k nearest neighbours then
        flag x for removal
for all x ∈ T do
    if x flagged for removal then T = T − {x}
> Iterate until no cases flagged for removal:
repeat
    for all x ∈ T do
        compute reachable(x)
        compute coverage(x)
    progress = false
    for all x ∈ T do
        if |reachable(x)| > |coverage(x)| then
            flag x for removal
            progress = true
    for all x ∈ T do
        if x flagged for removal then T = T − {x}
until not progress
return T
```

Figure 6. The Iterative Case Filtering Algorithm. First perform a noise filtering pass, then iteratively remove all cases with a larger reachable set than coverage set.

The remainder of the algorithm concentrates on removing superfluous cases in the manner described above. A check is carried out to make sure progress is being made after each iteration. The algorithm is decremental in nature, like the RT algorithms, but it differs in that more than one pass is required to thin the dataset.

We evaluated the ICF algorithm on 30 datasets[1] taken from the UCI repository of machine learning databases (Blake and Merz, 1998). The maximum number of iterations performed, of the 30 datasets, was 17. This number of iterations was required for the switzerland database, where the algorithm removed an average of 98% of cases. However, a number of the datasets consistently require as little as 3 iterations. Examining each iteration of the algorithm, specifically the percentage of cases removed after each iteration, provides us with an important insight into how the algorithm is working. We call this the reduction profile and is a characteristic of the case-base. Of the 30 datasets used, we isolated the two extreme reduction profiles which can be seen in figure 7. These were found for the switzerland database and the zoo database. The switzerland database exhibits a slow path to convergence. On average, a maximum of 17 iterations are required, each one removing at most 13% of the case-base and at minimum 2% of the case-base. The zoo database, on the other hand, exhibits fast convergence. An average of two iterations is required, with an average of 37% of cases being removed on the first pass.

By examining how many cases are removed after each iteration, we can imagine the possible nature of the case-base structure. For example, with the switzerland database many iterations are required, with a small number of cases being removed each time. This

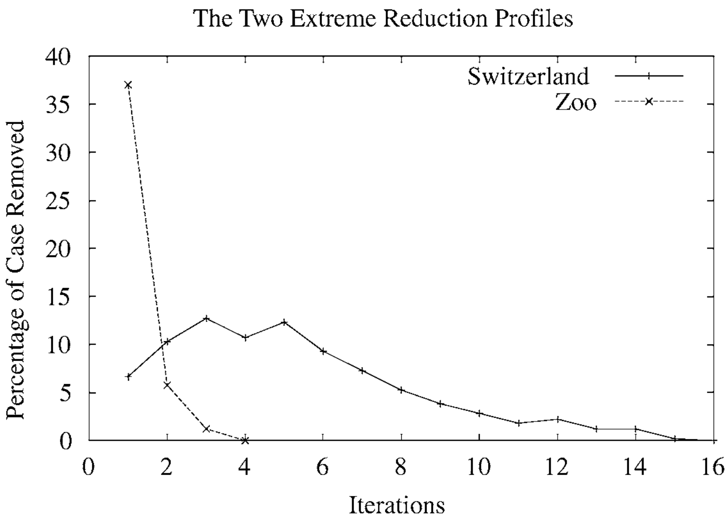


Figure 7. The two most extreme reduction profiles resulting from the ICF algorithm when it was applied to 30 domains. The switzerland database is hard to filter, and this is represented by a long series of iterations with each iteration removing a small proportion of the instances. Contrast this with the zoo database which requires an average of only 3 iterations, and on average, 37% of cases are removed after the first iteration.

observation would indicate that a high proportion of inter-related regions exist, as in order for one region to be thinned, a series of others must be thinned first. The length of the series reflects the complexity, rather than the size of the regions being filtered. Profiles exhibiting a short series of iterations, each one removing a large number of cases, would indicate a simple case-base structure containing little inter-dependency between regions. The most problematic of case-base structures would be characterised by a long series of iterations which results in few cases being removed.

3.4. Summary

We have discussed the principle approaches to instance set reduction devised over the last 30 years. The early schemes either address the problem of competence enhancement or competence preservation, but not both. The more recent approaches tend to attack both problems. The majority of methods aimed at preserving competence do not explicitly try and retain border cases, but rather use classification trials to see if the instance is useful. This is more a feedback driven model in comparison to the other methods, which tend to use criteria dependent on presuppositions about structure of the class definitions. In the next section we discuss work which compares these approaches.

Throughout this review our chief concern is with the case-removal criterion used by each algorithm. There are important practical considerations in how an algorithm processes the training set. With the exception of Aha's work, the algorithms discussed above rely on having a batch of cases on which to base case removal decisions: some of the measures used rely on examining all the cases in the training set. Aha's algorithms are incremental: they build up a concept description case-by-case rather than examining all the cases at once. It is important to note that with large databases, batch processing in the terms presented here may not be possible as holding all the cases in memory might not be practical. However, one could split a large database into manageable chunks, and process each chunk. We have chosen not to pursue these issues as our chief concern is with the diversity and utility of different selection criteria.

4. Comparative evaluation

Throughout the long development of instance pruning schemes one problem persists: little comparative evaluation between methods has been carried out, and those that have are not experimentally consistent with each other. To a degree, this problem still persists. In this section we aim to provide a comparison of the methods. Much of the work is already done by Wilson and Martinez (1997) and ourselves (Brighton and Mellish, 1997, 1999). For the purposes of comparison it is useful to group the approaches into three chronological groups:

1. Early approaches: CNN, RNN, SNN, Chang, Wilson Editing, Repeated Wilson Editing, and All k-NN.
2. Recent additions: IB2, IB3, TIBLE, Cameron-Jones's Extensions (Cameron-Jones, 1992).
3. State of the Art: RT3, ICF.

Roughly speaking, these three groups also encapsulate three classes of performance. Wilson and Martinez (1997) compared many of the early approaches with the recent additions, as well as RT3. Wilson and Martinez found RT3 to be consistently superior over 30 different domains. Brighton and Mellish (1999) carried out an similar study comparing RT3 with the ICF algorithm. In Brighton and Mellish (1997) we also compared the ICF and RT3 algorithm with some of those algorithms drawn from the early approaches. Our results agreed with those of Wilson and Martinez. Given this evidence it is apparent that progress has been made despite the lack of comparison: performance has got progressively better, we are closer to achieving our goal of unintrusive storage reduction . In this section we will concentrate on the comparison between RT3 and ICF. The reader is referred to the article by Wilson and Martinez for their comparison between the early methods, the recent additions, and RT3.

4.1. ICF versus RT3

Comparing the ICF algorithm with RT3, the most successful of Wilson and Martinez's algorithms, we found that the average case behaviours over the 30 datasets were very similar (See Table 1). Neither algorithm consistently outperformed the other. Both algorithms narrowly achieved an average case generalisation accuracy greater than that of the basic nearest-neighbour classifier. Both algorithms achieved approximately 80% reduction over the 30 domains. More interestingly, the behaviour of the two algorithms differ considerably on some problems. We find that no one deletion criterion consistently wins out. If we refer

Table 1. The classification accuracy and storage requirements for each dataset. The benchmark competence, which is the accuracy achieved without any filtering, is compared with Wilson Editing, RT3, and ICF.

	Benchmark		Wilson editing		RT3		ICF	
Dataset	Acc.	Stor.	Acc.	Stor.	Acc.	Stor.	Acc.	Stor.
abalone	19.53	100	22.01	19.64	22.11	40.95	22.74	15.11
anneal	95.28	100	93.24	95.46	91.82	20.72	91.35	22.59
balance-scale	77.36	100	86.04	77.48	83.40	18.23	81.47	14.67
breast-cancer-w	95.76	100	96.33	95.56	95.26	3.13	95.14	4.27
breast-cancer-l	62.46	100	68.42	64.69	74.42	19.94	72.81	23.51
bupa	59.71	100	61.81	60.49	61.23	35.07	60.75	24.79
cleveland	77.67	100	78.67	77.39	78.89	20.92	72.08	15.60
credit	82.32	100	84.46	81.12	83.15	19.9	82.28	16.89
ecoli	81.94	100	86.27	81.77	82.84	15.76	81.34	14.06
fleas	100.00	100	99.64	100.00	98.21	19.64	98.21	30.28
glass	71.43	100	69.05	70.17	69.05	23.26	69.64	31.40
hepatitis	85.16	100	82.10	84.48	83.33	19.15	82.26	16.33
hungarian	76.55	100	79.91	77.03	80.17	9.81	78.30	12.15
iris	95.00	100	95.33	96.21	93.61	16.04	92.56	42.08
led	63.77	100	68.27	66.11	69.62	18.04	71.74	41.81
led-17	42.82	100	43.00	43.09	41.48	46.78	42.33	27.50
lymphography	77.59	100	76.38	79.41	72.70	26.73	77.59	25.63
mushrooms	99.92	100	99.24	99.64	98.89	5.50	98.64	12.80
pima-indians	69.54	100	71.27	69.20	71.08	22.38	69.17	17.22
post-operative	57.78	100	66.94	54.65	69.44	6.45	65.28	7.18
primary-tumor	36.57	100	36.57	35.81	39.43	30.76	37.06	18.32
switzerland	92.08	100	93.54	90.45	91.67	2.15	92.28	2.02
thyroid	90.93	100	89.30	91.48	77.91	16.23	86.63	21.85
voting	92.99	100	93.28	92.76	93.77	7.43	91.19	8.88
waveform	75.36	100	76.62	76.37	76.14	22.79	73.93	18.98
wine	84.57	100	86.43	85.17	86.43	15.37	83.81	12.00
wisconsin-bc-di	93.01	100	93.85	92.94	92.92	6.95	92.99	6.38
wisconsin-bc-pr	67.18	100	75.90	72.64	76.28	15.43	75.64	18.24
yeast	52.70	100	55.39	52.97	55.32	27.03	52.25	16.62
zoo	95.50	100	96.25	95.31	87.08	26.13	92.42	52.78
Average	75.75	100	77.52	75.98	76.59	19.29	76.13	19.73

back to the theoretical limits discussed in Subsection 2.1, we notice that this is exactly what our average case results should look like. In our experiments, we retain 20% of the instances for testing, which means that (theoretically) only 20% of the training set is required to achieve competence preservation, and this is what we achieve in the average case.

We also found that the domains which suffer a competence degradation as a result of filtering using ICF and RT3 are exactly those for which competence degrades as a result of noise removal. This would indicate that noise removal is sometimes harmful, and both ICF and RT3 suffer as a consequence. This result supports the conclusions of Daelemans et al. (1999) who argue that filtering natural language problems is unwise due to the number of class exceptions. Class exceptions in the domains we consider would appear as noise to the filters that we employ, and would therefore be removed. However, Daelemans et al. do not use a filtering criterion that sufficiently ensures the retention of border cases, so the only real conclusion we can draw is that noise removal is unwise when datasets contain many class exceptions, rather than filtering in general. This does not bode well when we consider our objective of finding a consistent case filtering criteria: the characterisation of noise in some domains will be analogous to the characterisation of class exceptions in other domains. If the exceptions are single case exceptions, then it is impossible to differentiate.

To summarise, we have presented an algorithm which iteratively filters a case-base using an instance-based learning parallel of the two case properties used in the CBR Footprint Deletion policy. Due to the iterative nature of the algorithm, we have gained an insight into how the deletion of regions depend on each other. The point at which our deletion criterion ceases to hold (quite elegantly) results in unintrusive storage reduction. Our algorithm rivals the most successful scheme of those devised by Wilson and Martinez. Our results indicate that in some problems, noise cannot be differentiated from class exceptions.

5. Conclusions

We began by outlining some practical issues. We argued that different domains can sometimes have drastically different class structures and classified these domains into those with either homogeneous or non-homogeneous class structures. This is important as the notion of an instance critical to the classification process depends on this distinction. In the field of data mining homogeneous class definitions are the norm, and this article concentrates on those schemes that perform instance selection on these problems.

After reviewing the principle approaches we grouped them into three classes: early schemes, recent additions, and the state of the art. The degree to which each class of algorithm achieves unintrusive storage reduction approximately mirrors this chronological order. We found that our ICF algorithm and Wilson and Martinez' RT3 algorithm achieve the highest degree of instance set reduction as well as the retention of classification accuracy: they are close to achieving unintrusive storage reduction. The degree to which these algorithms perform is quite impressive: an average of 80% of cases are removed and classification accuracy does not drop significantly. The comparison we provide is important as, considering the number of approaches, few consistent comparisons have been made. In our review we also direct the reader to the work located on the fringes of this area.

The chief point we wish to address is that, traditionally, reduction schemes have been seen as general solutions to the problem of instance selection. Our observations on how these schemes work, and how well they work in different problems, suggest that the success of a scheme is highly dependent on the structure of the instance-space. We argue that one selection criterion is not enough for high performance across the board. Our results reinforce this point, especially when we consider that the problem coverage in our experiments is minimal in comparison to the variety of databases we might encounter. If we look at larger and more complex datasets, the point is likely to reinforced still. Similarly, in the context of noise removal, problem specific dependency is also a problem. We do not have a full understanding of the problem dependency, but the reduction profile provided by our ICF algorithm is a first step in achieving more perspicuous insights into problem structure. For example, some domains may contain both homogeneous and non-homogeneous class structures, in which case we have a problem because certain parts of the instance space are best served by different reduction criteria: both prototypical and border cases are required for the most effective solution. The local-set construction we introduced in Section 3.2 could also be used as a measure of how homogeneous the class structures in instance space are. By computing the average local-set size, we would have a measure of how local instances of the same class are to each other. For example, an instance space with a low average local-set size might either contain lots of noise, or plenty of class exceptions.

To summarise, we argue that with the majority of classification problems, border cases are critical to discrimination between classes. We introduce an algorithm which rivals the most successful existing algorithm over 30 domains. The result of comparison of these algorithms is that neither is consistently superior. Deployment is the issue if we wish to ensure successful instance selection, and the key to deployment is the insights we have into how classes are constructed within the instance space.

Acknowledgments

Richard Nock, David Willshaw, and Pete Whitelock provided invaluable advice during the preparation of this work. The three anonymous reviewers also helped to clarify the work, as well as pointing out areas for improvement. We would also like to thank both the donors and maintainers of the UCI Repository for Machine Learning Databases.

Note

1. The results presented here represent the average case statistics over 20 independent runs. A random 20% of cases where taken to form a testing set. Different testing sets where taken for each run. All feature values where normalised, with non-ordered symbolic features transformed to a feature vectors. 1-nearest neighbour classification was used throughout.

References

Aha, D.W., Kibler, D., and Albert, M.K. 1991. Instance based learning algorithms. Machine Learning, 6(1):37–66.

Blake, C. and Merz, C. 1998. UCI repository of machine learning databases.

Brighton, H. 1996. Experiments in case-based learning. Undergraduate Dissertation, Department of Artificial Intelligence, University of Edinburgh, Scotland.

Brighton, H. 1997. Information filtering for lazy learning algorithms. Masters Thesis, Centre for Cognitive Science, University of Edinburgh, Scotland.

Brighton, H. and Mellish, C. 1997. Geometric criteria for case deletion in case-based learning algorithms. Unpublished article.

Brighton, H. and Mellish, C. 1999. On the consistency of information filters for lazy learning algorithms. In Principles of Data Mining and Knowledge Discovery, 3rd European Conference, Prague, Czech Republic, J.M. Zytkow and J. Rauch (Eds.), pp. 283–288.

Brodley, C. 1993. Addressing the selective superiority problem: Automatic algorithm/mode class selection. In Proceedings of the Tenth International Machine Learning Conference, Amherst, MA, pp. 17–24.

Cameron-Jones, R.M. 1992. Minimum description length instance-based learning. In Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence, Hobart, Australia, pp. 368–373.

Chang, C.-L. 1974. Finding prototypes for nearest neighbor classifiers. IEEE Transactions on Computers, C-23:1179–1184.

Cover, T.M. and Hart, P.E. 1967. Nearest neighbor pattern classification. IEEE. Transactions on Information Theory, IT-13:21–27.

Daelemans, W., van den Bosch, A., and Zavrel, J. 1997. A feature-relevance heuristic for indexing and compressing large case bases. In Proceedings of the 9th European Conference on Machine Learning, Prague, Czech Republic, pp. 29–38.

Daelemans, W., van den Bosch, A., and Zavrel, J. 1999. Forgetting exceptions is harmful in language learning. Machine Learning, 34(1/3): 11–41.

Dasarathy, B. 1991. Nearest Neighbor (NN) norms: NN Pattern Classification Techniques. Los Alimos, CA: IEEE Computer Society Press.

Domingos P. 1995. Rule induction and instance-based learning: A unified approach. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Montreal, Canada, pp. 1226–1232.

Gates, G.W. 1972. The reduced nearest neighbor rule. IEEE Transactions on Information Theory, 18(3):431–433.

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

Holte, R.C., Acker, L., and Porter, B. 1989. Concept learning and problem of small disjuncts. In Proceedings of the 11th International Joint Conference on Artificial Intelligence, pp. 813–818.

King, R.D., Feng, C., and Sutherland, A. 1995. Statlog: Comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence, *9*(3):289–333.

Kolodner, J.L. 1993. Case-Based Reasoning. San Mateo, CA: Morgan Kaufmann.

Markovitch, S. and Scott, P.D. 1988. The role of forgetting in learning. In Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI, pp. 459–465.

Markovitch, S. and Scott, P.D. 1993. Information filtering: Selection mechanisms in learning systems. Machine Learning, 10(2):113–151.

Ritter, G.L., Woodruff, H.B., Lowry, S.R., and Isenhour, T.L. 1975. An algorithm for the selective nearest neighbour decision rule. IEEE Transactions on Information Theory, 21(6):665–669.

Salganicoff, M. 1993. Density-adaptive learning and forgetting. In Proceedings of the 10th International Conference on Machine Learning, University of Massachusetts, Amherst, pp. 276–283.

Salzberg, S. 1991. A nearest hyperrectangle learning method. Machine Learning, 6:227–309.

Sebban, M., Zighed, D.A., and Di Palma, S. 1999. Selection and statistical validation of features and prototypes. In Principles of Data Mining and Knowledge Discovery, 3rd European Conference. Prague, Czech Republic, J.M. Zytkow and J. Rauch (Eds.), pp. 184–192.

Smyth, B. and Keane, M.T. 1995. Remembering to forget. In IJCAI-95, Proceedings of the Fourteenth International Conference on Artificial Intelligence, C.S. Mellish (Ed.), Vol. 1., pp. 377–382.

Swonger, C.W. 1972. Sample set condensation for a condensed nearest neighbour decision rule for pattern recognition. In Frontiers of Pattern Recognition, S. Watanabe (Ed.), Orlando, FA: Academic Press, pp. 511–519.

Tomek, I. 1976. An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(6):448–452.

van den Bosch, A. and W. Daelemans, 1998. Do not forget: Full memory in memory-based learning of word pronunciation. In Proceedings of NeMLaP3/CoNLL98, Sydney, Australia, pp. 195–204.

Wilson, D.L. 1972. Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics, SMC-2(3):408–421.

Wilson, D.R. and Martinez, A.R. 1997. Instance pruning techniques. In Machine Learning: Proceedings of the Fourteenth International Conference, D. Fisher (Ed.). San Francisco, CA.

Zhang, J. 1992. Selecting typical instances in instance-based learning. In proceedings of the Ninth International Machine Learning Conference, Aberdeen, Scotland, pp. 470–479.

Henry Brighton is a PhD student at the Language Evolution and Computation Research Unit in the Department of Linguistics, the University of Edinburgh. Previously he worked for Sharp Laboratories of Europe in the field of Machine Translation. His research interests include all forms of machine learning, but specifically, how different learning biases impact on the cultural evolution of language.

Chris Mellish is a Reader in the Division of Informatics at the University of Edingburgh. He received his PhD in Artificial Intelligence from Edinburgh. He has worked mainly in the fields of Natural Language Processing and Logic Programming. He is especially interested in Natural Language Generation. He has co-authored textbooks on Prolog and Natural Language Processing.