On the consistency of information filters for lazy learning algorithms

Henry Brighton[1] and Chris Mellish[2]

[1] SHARP Laboratories of Europe Ltd., Oxford Science Park, Oxford, England, UK
henry@sharp.co.uk
[2] Department of Artificial Intelligence, The University of Edinburgh, Scotland, UK
chrism@dai.ed.ac.uk

Abstract. A common practice when filtering a case-base is to employ a filtering scheme that decides which cases to delete, as well as how many cases to delete, such that the storage requirements are minimized and the classification competence is preserved or improved. We introduce an algorithm that rivals the most successful existing algorithm in the average case when filtering 30 classification problems. Neither algorithm consistently outperforms the other, with each performing well on different problems. Consistency over many domains, we argue, is very hard to achieve when deploying a filtering algorithm.

1 Introduction

Information filtering is an attractive proposition when working with lazy learning algorithms. The lazy learning paradigm is characterized by the indiscriminate storage of training cases during the training stage. To classify an unseen query case a lazy learner applies the nearest neighbor algorithm [4]. We focus on how the size of the database holding the cases can be minimised such that the classification response time can be improved. By removing harmful cases we can also increase the overall classification competence of the learner. In this paper we introduce a new algorithm for filtering case-bases used by lazy learning algorithms. After comparing the algorithm with the most successful existing filter on 30 datasets from the UCI repository for machine learning databases [5], we conclude that neither approach is consistently superior.

2 Issues in case filtering

By removing a set of cases from a case-base the response time for classification decisions will decrease, as fewer cases are examined when a query case is presented. The removal can also lead to either an increase or decrease in classification competence. Therefore, when applying a filtering scheme to a case base we must be clear about the degree to which we are willing to let the original classification competence depreciate. Typically, the principle objective of a filtering

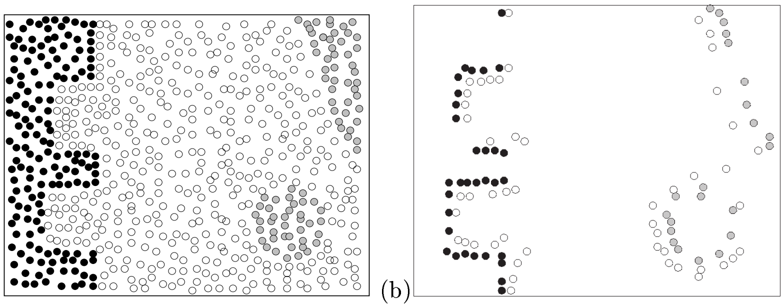
(a)  (b)

Fig. 1. (a) The 2d-dataset. (b) The cases remaining from the 2d-dataset after 5 iterations of the ICF algorithm.

algorithm is unintrusive storage reduction. Here, classification competence is primary: we desire the same (or higher) learning competence but we require it faster and taking up less space. Ideally, competence should not suffer at the expense of improved performance. If our filtering decisions are not to harm the classification competence of the learner, we must be clear about the kind of deletion decisions that *introduce* misclassifications. Consider the following reasons why a k-nearest neighbor classifier might misclassify an unseen query case:

1. When noise is present in locality of the query case. The noisy cases(s) win the majority vote, resulting in the incorrect class being predicted.
2. When the query case 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 cases 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 a lazy learner. This may be due to the nature of the underlying function, or due to the sparse data problem.

In the context of filtering, we can address point (1) and try and improve classification competence 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 cases close to borders is not recommended as these cases are relevant to discrimination between classes. We should be aware of point (3), but as k is typically small, the occurence of such a problem is likely to be rare. Consider our example dataset shown in Figure 1(a), we can imagine that removing the interior of the class regions would not lead to a misclassification of a query case at these points: the border cases still supply the required information.

3 Review

Filtering the set of stored instances has been an issue since the early work on nearest neighbor (NN) classification [4]. The early schemes typically concentrate on either competence enhancement (noise removal)[8] or competence preservation [6]. More recent schemes attempt both [1, 9].

A novel approach to competence preservation is the *Footprint Deletion* policy of Smyth and Keane [6] which is a filtering scheme designed for use within the paradigm of Case- Based Reasoning (CBR). In previous work [3] we have shown that some of the concepts introduced by Smyth and Keane transfer to the simpler context of lazy learning. 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 Lazy Learning parallel termed the *Local-Set* of a case c to capture this property [2]. We define the *Local-set* of a case c as: *The set of cases contained in the largest hypersphere centered on c such that only cases in the same class as c are contained in the hypersphere.*

The novelty 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 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 Lazy 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 neighbors 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 neighborhood of c' by the first case of a differing class. Armed with this parallel we found that Footprint deletion performed well [2]. 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. Local-set deletion has subsequently been employed in the context of natural language processing [7].

```
ICF(T)
1   /* Perform Wilson Editing */
2   for all x ∈ T do
3       if x classified incorrectly by k nearest neighbours then
4           flag x for removal
5   for all x ∈ T do
6       if x flagged for removal then T = T − {x}
7   /* Iterate until no cases flagged for removal */
8   repeat
9       for all x ∈ T do
10          compute reachable(x)
11          compute coverage(x)
12      progress = false
13      for all x ∈ T do
14          if |reachable(x)| > |coverage(x)| then
15              flag x for removal
16              progress = true
17      for all x ∈ T do
18          if x flagged for removal then T = T − {x}
19  until not progress
20  return T
```

Fig. 2. The Iterative Case Filtering Algorithm.

4 An iterative case filtering algorithm

We now present a new algorithm which uses an iterative approach to case deletion. We term the algorithm the *Iterative Case Filtering Algorithm* (ICF). The ICF algorithm uses the lazy learning parallels of case coverage and reachability we developed when transferring the CBR footprint deletion policy, discussed above. 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 *neighborhood* and *associate* sets used by [9]. 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. After removing these cases the case-space will typically contain thick bands of cases either side of class borders. The algorithm is depicted in Figure 2. We also employ the noise filtering scheme based on Wilson Editing and adopted by [9]. Lines 2-6 of the algorithm perform this task. Figure 1(b) depicts the 2d-dataset, introduced earlier, after 5 iterations of the

ICF algorithm. 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 recomputing, the criterion no longer holds.

We evaluated the ICF algorithm on 30 datasets found at the UCI repository of machine learning databases [5]. 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. 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 resulting in few cases being removed. Comparing the ICF algorithm with RT3, the most successful of Wilson and Martinez's algorithms, we found that the average case behaviors over the 30 datasets were very similar (See Table 1). Neither algorithm consistently outperformed the other. More interestingly, the behavior of the two algorithms differ considerably on some problems. We also found that the domains which suffer a competence degradation as a result of filtering using ICF and RT3 are exactly those in 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. To summarize, we have presented an algorithm which iteratively filters a case-base using a lazy 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 seases to hold can result in improved generalization accuracy and storage reduction.

5 Conclusion

We have introduced the ICF algorithm which supports the argument that consistency is hard. We compared the ICF algorithm with a recent successful algorithm RT3 and found that their average case performance is very similar, but on individual problems they can differ considerably. Each algorithm can outperform the other on certain problems, both in terms of competence and storage reduction. Consistency is therefore a problem in the deployment of filtering schemes. One advantage of our algorithm is that it provides us with a reduction profile. The profile tells how different regions are dependent on each other. This provides us with a useful degree of perspicuity in understanding the structure of the case-base. Utimately, the choice of filter we deploy must be informed by any insights we have into the structure of the case-space.

Table 1. The classification accuracy and storage requirements for a sample of the datasets mentioned. The benchmark competence, which is the accuracy acheived whithout 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
balance-scale	77.36	100	86.04	77.48	83.40	18.23	81.47	14.67
cleveland	77.67	100	78.67	77.39	78.89	20.92	72.08	15.60
ecoli	81.94	100	86.27	81.77	82.84	15.76	81.34	14.06
glass	71.43	100	69.05	70.17	69.05	23.26	69.64	31.40
hungarian	76.55	100	79.91	77.03	80.17	9.81	78.30	12.15
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
pima-indians	69.54	100	71.27	69.20	71.08	22.38	69.17	17.22
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
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
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

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
2. H. Brighton. Experiments in case-based learning. Undergraduate Dissertation, Department of Artificial Intelligence, University of Edinburgh, Scotland, 1996.
3. H. Brighton. Information filtering for lazy learning algorithms. Masters Thesis, Centre for Cognitive Science, University of Edinburgh, Scotland, 1997.
4. T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, IT-13:21 – 27, 1967.
5. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1996. Irvine, CA: University of California, Department of Information and Computer Science.
6. B. Smyth and M. T. Keane. Remembering to forget. In C. S. Mellish, editor, *IJCAI-95: Proceedings of the Fourteenth International Conference on Artificial Intelligence*, volume 1, pages 377 – 382. Morgan Kaufmann Publishers, 1995.
7. Antal van den Bosch and Walter Daelemans. Do not forget: Full memory in memory-based learning of word pronunciation. In *Proceedings of NeMLaP3/CoNLL98*, pages 195 – 204, Sydney, Australia, 1998.
8. D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3):408 – 421, Jun 1972.
9. D. R. Wilson and A. R. Martinez. Instance pruning techniques. In D. Fisher, editor, *Machine Learning: Proceedings of the Fourteenth International Conference*, San Francisco, CA, 1997. Morgan Kaufmann.