Noise-Tolerant Windowing

Johannes Fiirnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
E-mail: juffi@ai.univie.ac.at

Abstract

Windowing has been proposed as a procedure for efficient memory use in the ID3 decision tree learning algorithm. However, it was shown that it may often lead to a decrease in performance, in particular in noisy domains. Following up on previous work, where we have demonstrated that the ability of rule learning algorithms to learn rules independently can be exploited for more efficient windowing procedures, we demonstrate in this paper how this property can be exploited to achieve noise-tolerance in windowing.

1 Introduction

Windowing is a general technique that aims at improving the efficiency of inductive classification learners. The gain in efficiency is obtained by identifying an appropriate subset of the given training examples, from which a theory of sufficient quality can be induced. Such procedures are also known as *subsampling.* Windowing has been proposed in [Quinlan, 1983] as a supplement to the inductive decision tree learner ID3 to enable it to tackle tasks which would otherwise have exceeded the memory capacity of the computers of those days.

Despite first successful experiments in the KRKN chess endgame domain [Quinlan, 1983], windowing has not played a major role in machine learning research. One reason for this certainly is the rapid development of computer hardware, which made the motivation for windowing seem less compelling. However, recent work in the areas of *Knowledge Discovery in Databases* [Kivinen and Mannila, 1994; Toivonen, 1996] *and Intelligent Information Retrieval* [Lewis and Catlett, 1994; Yang, 1996] has recognized the importance of subsampling procedures for reducing both, learning time and memory requirements.

A good deal of this lack of interest can be attributed to an empirical study [Wirth and Catlett, 1988] which showed that windowing is unlikely to gain any efficiency. The authors studied windowing with ID3 in various domains and concluded that it cannot be recommended as a procedure for improving efficiency. The best results were achieved in noise-free domains, such as the *Mushroom* domain, where it was able to perform on the same level as ID3 without windowing, while its performance in noisy domains was considerably worse. In [Fiirnkranz, 1997a], we have demonstrated that rule learning algorithms are better suited for windowing in noise-free domains, because they learn each rule independently. In this paper, we will show how this property can be exploited in order to achieve noise-tolerance.

2 The I-RIP algorithm

We have conducted our study in the framework of *separate-and-conquer* rule learning algorithms that has recently gained in popularity [Fiirnkranz, 1997b]. The basic learning algorithm we use, I-RIP, is based on I-REP [Fiirnkranz and Widmer, 1994] and its successor RIPPER [Cohen, 1995]. However, the algorithms presented in this paper do not depend on this choice; any other effective noise-tolerant rule learning algorithm could be used in I-RIP's place.

I-REP achieves noise-tolerance by first learning a single, consistent rule on two thirds of the training data and then pruning this rule on the remaining third. The resulting rule is added to the theory, and all examples that it covers are removed from the training set. The remaining training examples are used for learning another rule until no more meaningful rules can be discovered. In [Cohen, 1995] it was shown that some of the parameters of the 1-REP algorithm, like the pruning and stopping criteria, were not chosen optimally. We have implemented the I-REP algorithm as described in [Fiirnkranz and Widmer, 1994], but used RIPPER's rule-value-metric pruning criterion and its 0.5-rule-accuracy stopping criterion. We have not implemented RIPPER's rule optimization heuristics. Thus our I-RIP algorithm is half-way between I-REP and RIPPER. As such, it is quite similar to I-REP*, which is also described in [Cohen, 1995], but it differs from it in that its implementation is closer to the original I-REP. For example, I-RIP considers every condition in a rule for pruning, while I-REP* only considers to delete a final sequence of conditions. On the other hand, I-REP* is able to handle numerical variables, missing values, and multiple classes, which our implementation of I-RIP currently does not support. However, these are no principle limitations to the algorithm, and standard enhancements for dealing with these problems could easily be added to all algorithms described in this paper.

3 Windowing and Noise

The windowing algorithm described in [Quinlan, 1983] starts by picking a random sample of a user-settable size *InitSize* from the total set of *Examples* and uses it for inducing a classifier with a given learning algorithm, in our case the I-RIP algorithm briefly described in the last section. This theory is then tested on the remaining examples and the examples that it misclassifies are moved from the test set to the window. Another parameter, *MaxIncSize,* aims at keeping the window size small. If this number is exceeded, no further examples are tested and the next iteration starts with the new window. To ensure that all examples are tested in the first few iterations, our implementation takes care that those examples which remain untested in one iteration will be tested first in the subsequent iteration. We have named our implementation of a windowed version of I-RIP WIN-RIP.

An efficient adaptation of this windowing technique to noisy domains is a non-trivial endeavor. In particular, it cannot be expected that the use of a noise-tolerant learning algorithm like I-RIP inside the windowing loop will lead to performance gains in noisy domains. The contrary is true: the main problem with windowing in noisy domains lies in the fact that it will eventually incorporate all noisy examples into the learning window, because they will be misclassified by a good theory. On the other hand, the window will typically only contain a subset of the original learning examples. Thus, after a few iterations, the proportion of noisy examples in the learning window can be much higher than the noise level in the entire data set, which will make learning considerably harder.

Assume for example that WIN-RIP has learned a correct theory from 1000 examples in a 11,(XX) examples domain, where 10% of the examples are misclassified due to noise. In the next iteration, about 1000 noisy examples will be misclassified by the correct theory and will be added to the window, thus doubling its size. Assuming that the original window also contained about 10% noise, more than half of the examples in the new window are now erroneous, so that the classification of the examples in the new window is in fact mostly random. It can be assumed that many more examples have to be added to the window in order to recover the structure that is inherent in the data. This hypothesis is consistent with the results of [Wirth and Catlett, 1988] and iCatlett, 1991], where it was shown that windowing is highly sensitive to noise.

4 A Noise-Tolerant Version of Windowing

The windowing algorithm described in [Furnkranz, 1997a], which is only applicable to noise-free domains, is based on the observation that rule learning algorithms will re-discover good rules again and again in subsequent iterations of the windowing procedure. Such consistent rules do not add examples to the current window, but they nevertheless have to be re-discovered in subsequent iterations. If these rules could be detected early on, they could be saved and the examples they cover could be removed from the window, thus gaining computational efficiency. The algorithm discussed in [Ftumkranz, 1997a] achieves this by separating the examples that are covered by rules that have been consistent for a larger number of examples, so that subsequent iterations only have to learn rules for the yet uncovered parts of the search space.

```
procedure I-WIN(Examples,InitSize,MaxIncSize)

Train = RandomSample(Examples,InitSize)
Theory = $\emptyset$
repeat
    NewTheory = I-RIP(Train)
    NewTr = Train
    NewEx = Examples
    Candidates = $\emptyset$
    for Rule $\in$ NewTheory
        if Significant(Rule,Examples)
            Theory = Theory $\cup$ Rule
            NewTr = NewTr $\setminus$ Cover(Rule,Train)
            NewEx = NewEx $\setminus$ Cover(Rule,Examples)
        else
            Candidates = Candidates $\cup$ Cover(Rule,Examples)
    for Example $\in$ Positive(Examples)
        if Example $\notin$ Cover (NewTheory,Examples)
            Candidates = Candidates $\cup$ Example
    Train = NewTr $\cup$ RandomSample(Candidates,MaxIncSize)
    Examples = NewEx
until Candidates = $\emptyset$
return (Theory)
```

Figure 1: A noise-tolerant version of windowing.

The 1-WIN algorithm shown in figure 1 is based on the same idea. At the beginning the algorithm proceeds just like WIN-RIP: it selects a random subset of the examples, learns a theory from these examples, and tests it on the remaining examples. However, contrary to WIN-RIP, it does not merely add examples that have been incorrectly classified to the window for the next iteration, but it also removes all examples from this window that are covered by good rules. To determine good rules, WIN-RIP tests the individual rules that have been learned from the current window on the entire data set and computes some quality measure from this information (procedure SIGNIFICANT in figure 1).

In principle, this quality measure is a parameter of the windowing algorithm. For example, one could use a measure as simple as "consistency with the negative examples" in order to get a windowing algorithm that is suitable for learning from noise-free data sets. However, in noisy domains, noise-tolerant learning algorithms will typically produce rules that are not consistent with the training data. Thus, a more elaborate criterion must be used. We have experimented with a variety of criteria known from the literature, but found that they are insufficient for our purposes. For example, it turned out that, at higher training set sizes, CN2's likelihood ratio significance test [Clark and Niblett, 1989] will deem almost any rule learned by I-RIP as significant, even if the distribution of covered positive and negative examples deviates only slightly from their distribution in the entire training set.

Eventually, we have settled for the following criterion: For each rule r learned from the current window we compute two accuracy estimates, *AccWin(r)* which is determined using only examples from the current window and *AccTot(r)* which is estimated on the entire training set. The criterion we use for detecting good rules consists of two parts:

- The $AccWin(r)$ estimate has to be significantly above the default accuracy of the domain. This is ensured by requiring that $AccWin(r) - SE(AccWin(r)) > DA$, where DA is the default accuracy, and $SE(p) = \sqrt{\frac{p(1-p)}{n}}$ is the standard error of classification.
- The second criterion requires that $AccWin(r) - \alpha SE(AccWin(r)) \leq AccTot(r) \leq AccWin(r) + \alpha SE(AccWin(r))$, i.e., the estimate derived from the entire training set has to be in about the same range as the estimate derived from the learning sample. $\alpha \geq 0$ is a user-settable parameter that can be used to adjust the width of this range.

The purpose of the first heuristic is to avoid rules with a bad classification performance (in particular this weeds out many rules that have been derived from too few training examples), while the second criterion aims at making sure that the accuracy estimates that have been derived on the current window (and thus have been used in the heuristics of the learning algorithm) have not been too optimistic compared to the true accuracy of the rule, which is approximated by the accuracy measured on the entire training set.

The parameter α determines the degree to which the estimates $AccWin(r)$ and $AccTot(r)$ have to correspond. A setting of $\alpha = 0$ requires that $AccWin(r) = AccTot(r)$, which in general will only be true for consistent rules or for rules that have been learned from the entire training set. This is the recommended setting in noise-free domains. In noisy domains, values of $\alpha > 0$ have to be used, as the rules returned from the learning algorithm will typically be inconsistent on the training set. Note, however, that a setting of $\alpha = 0$ in a noisy domain will *not* lead to overfitting and a decrease in predictive accuracy, because it will not lead to the acceptance of consistent rules (which in general will not be returned by the noise-tolerant basic learning algorithm anyways), but only result in further growth of the window, until all relevant examples are in the window, in which case the rule will also be accepted by the second criterion. A typical setting in a noisy domain would be around $\alpha = 1$, but the parameter seems to be quite sensitive. $\alpha = \infty$ will move all rules that have survived the first criterion into the final rule set.

With a setting of $\alpha = 0$, I-WIN is very similar to the WIN-DOS-95 algorithm described in [Fürnkranz, 1997a] with the difference that WIN-DOS-95 only tests a theory until it has collected *MaxIncSize* new examples to add to the current window. Thus it cannot determine whether a rule has already been tested on all examples and has to test the stored rules in all subsequent iterations. I-WIN, on the other hand, tests a rule on the entire training set. If it finds the rule to be significant it will add it to the final rule set and will never test it again. Consequently, all examples covered by such a rule will be removed from the training set. If I-WIN finds the rule to be insignificant, all examples that are covered and not already contained in the current window will be considered as candidates for being added to the window. Positive examples that have not been covered by any of the rules will also be considered as such candidates. I-WIN randomly selects *MaxIncSize* of these candidate examples and adds them to the window. By sampling from *all* examples covered by-insignificant rules (not only negative examples as in regular windowing), we hope to avoid part of the problem outlined in the previous section. However, we stick to adding uncovered *positive* examples only, because after more and more rules have been discovered, the proportion of positive examples in the remaining training set will considerably decrease, so that the chances of picking one of them by random sampling will also decrease. Adding only positive uncovered examples may lead to over-general rules, but these will be discovered by the second part of our criterion and appropriate counter-examples will eventually be added to the window.

The actual implementation of our algorithm makes use of several optimizations that minimize the amount of testing that has to be performed in the algorithm. An important addition considers the case when the underlying learning algorithm is unable to learn any rules from the current window. Then, the algorithm in figure 1 will add *MaxIncSize* uncovered positive examples to the current window. Our implementation of the algorithm deals with these cases by doubling the window size and re-initializing it with a new random sample of the new size. We think that this may lead to faster convergence in some cases, but have not yet systematically tested this hypothesis. Furthermore, all algorithms discussed in this paper attempt to remove semantically redundant rules in a post-processing phase. Such rules only cover training examples that are also covered by other rules. We refer to [Furnkranz, 1997al for more details.

5 Experimental Evaluation

In each of the experiments described in this section, we report the average results of 10 different subsets of the specified training set size, selected from the entire set of preclassified examples. All algorithms were run on identical data sets, but some random variation may have resulted from the fact that I-RIP uses internal random splits of the training data. For each experiment we measured the accuracy of the learned theory on the entire example set and the total run-time of the algorithm.[1] All experiments shown below were conducted with a setting of *InitSize* = 100 and *MaxIncSize* = 50. These settings have been found to perform well on noise-free domains [Fiirnkranz, 1997a]. We have not yet made an attempt to evaluate their appropriateness for noisy domains.

First we have tested the algorithms on the 8124 example *Mushroom* database. Although this database is known to be noise-free, it forms an interesting test-bed for our algorithms, because it allows a rough comparison to previous results. For example, windowing with the decision tree learner ID3 could not achieve significant run-time gains over pure ID3 [Wirth and Catlett, 1988], while the slightly modified version of windowing used in C4.5 is able to achieve a run-time improvement of only about 15% (p. 59 of [Quinlan, 1993]).

The left column of figure 2 shows the accuracy and run-time results for I-RIP, WIN-RIP, and three versions of I-

[1] Measured in CPU seconds of a microSPARC 110MHz running compiled Allegro Common Lisp code under SUN Unix 4.1.3.

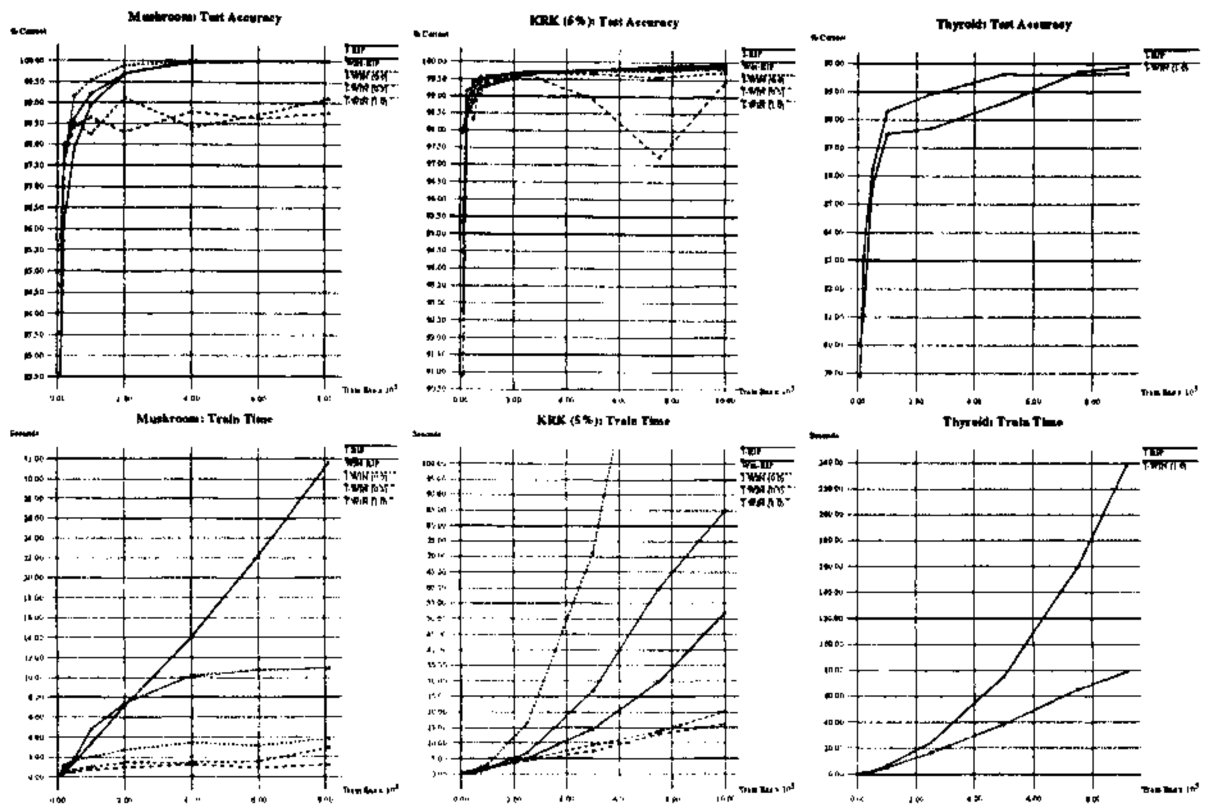

Figure 2: Accuracy and run-time results in the *Mushroom* domain, the *KRK* domain with 5% noise added, and a simplified version of the *Thyroid* domain.

WIN, each one using a different setting of its α parameter. In terms of run-time, both regular windowing, and our improved version are quite effective in this domain, at least for higher (> 1000) training set sizes. The three versions of I-WIN are clearly the fastest. In terms of accuracy, no significant differences can be observed between 1-RIP, WIN-RIP, and I-WIN (0.0), although the latter is able to compensate some of the weakness of I-RIP at low example set sizes that is due to its internal split of the data [FUrnkranz and Widmer, 1994]. 1-WIN with α - 0.5 and α = 1.0 has a significantly worse performance, because these versions are often content with slightly over-general rules, which is detrimental in this noise-free domain. However, we have shown that our windowing algorithm is in fact able to achieve significant gains in run-time without losing accuracy, thus confirming our previous results [FUrnkranz, 1997a].

For testing the algorithms' noise-handling capabilities we have performed a series of experiments in a propositional version of the well-known KRK classification task, which is commonly used as a benchmark for relational learning algorithms. The goal is to learn rules for recognizing illegal white-to-move chess positions with only the white king, the white rook, and the black king on the board. The propositional version of this domain consists of 18 binary attributes that encode the validity or invalidity of relations like adjacent, <, and = between the coordinates of three pieces on a chess board. We have generated 10,000 noise-free examples in this domain, which were always used for testing the accuracies of the learned theories. The training sets were generated by subsampling from the 10,000 example set. Artificial noise was generated by replacing the classification of *n%* of the training examples with a randomly selected classification (chosen with a fair coin). *Mushroom* domain, and are not shown here.

The middle column of figure 2 shows the results in the KRK domain at a very moderate noise level (5%). Regular windowing cannot achieve any performance gains. On the contrary, it is almost twice as expensive as I-RIP. I-WIN with a noise-free setting of $\alpha = 0$ is even more expensive: it needs more than 300 sees, for a 10,000 example training set, which is six times as much as I-RIP. The noise-tolerant versions of our algorithms outperform the other algorithms in terms of run-time. In terms of accuracy, a setting of $\alpha = 1.0$ seems to heavily over-generalize. $\alpha = 0.5$ performs reasonably well, although it is still a little behind in accuracy. The size of good values for α seems to have some correlation with the noise level in the data. We have performed experiments with various levels of noise and confirmed that higher values of *a* will produce better results with increasing levels of noise.[2]

In this domain, we also performed a series of experiments with the aim of analyzing the behavior of I-RIP and I-WIN over varying levels of artificial noise.[2] The results in terms of accuracy were very inconclusive with both algorithms having their ups and downs. In terms of run-time, we found that I-WIN outperforms I-RIP at lower noise levels, but the converse is true for higher noise levels. The more random the data are, the less likely it is that the rules learned by I-RIP from a window of small size will bear any significance. Thus I-WIN has to successively increase its window size without being able to remove any examples that are covered by rules learned in previous iterations. Consequently, it has much larger run-times than I-RIP, which learns only once from the entire data set. However, for reasonable noise levels, which can be expected to occur in most real-world applications (say < 30%), I-WIN significantly outperforms I-RIP. For example, 1-REP's run-time of 64.85 secs, for learning from the 10,000 example set with 10% noise is about 4 times higher than that of I-WIN with a setting of $\alpha = 1.0$. This advantage decreases with increasing noise-level: at a noise-level of 50%, I-WIN is still about 15% faster, but at 75% I-RIP is already about five times faster than I-WiN. The highest noise-level for which I-RIP is faster than I-WIN increases with training set size (5% for 1000 examples, 50% for 5000, 75% for 10000). We take this as evidence that the chances of I-WIN outperforming I-RIP increase with increasing training set sizes or with increasing redundancy in the data.

Currently, the implementation of our algorithms is limited to binary symbolic domains. The algorithms are not able to handle continuous attributes, missing values, or multiple classes, although nothing in the algorithms prevents the use of standard techniques for dealing with these problems, like the use of thresholds, turning multi-class tasks into a sequence of binary tasks, etc. Unfortunately, we were not able to detect a natural domain of a reasonable size in the UCI data repository which meets the constraints of our implementation. So we decided to try our algorithms on a discretized, 2-class version of Quinlan's 9172 example thyroid diseases database.[3] In this simplified domain, C4.5 without any pruning (the unpruned tree obtained with -m 1) achieves an accuracy of 88% (estimated by a 10-fold cross-validation) while the pruned tree obtained with default settings has an accuracy of 89.1 %. The respective tree sizes are 6570 vs. 181. We take this as evidence that the data set contains at least a moderate amount of noise. Consequently, C4.5's windowing procedure is quite inefficient and takes more than twice as long (> 40 CPU secs.) for growing a single tree from the entire data set (parameter -t 1) than C4.5 with default parameters (< 20 CPU secs.).

The right-most column of figure 2 shows the results in this domain. I-WIN with $\alpha = 1.0$ significantly outperforms I-RIP at both measures, run-time and accuracy. Only when the entire data set is used for both training and testing, I-RIP maintains an accuracy advantage. This, however, only raises the suspicion that I-RIP overfits the data in this domain, while the significance test used in I-WIN is able to correct this to some extent by evaluating the predictive performance of the simpler rules learned at low window sizes on the entire training set.

6 Further Research

I-WIN contains several parameters. In all experiments in this paper we have set the initial window size to 100, and the maximum window increment to 50. We have found these parameters to perform well on noise-free domains [Furnkranz, 1997a], but in some experiments we have encountered evidence that larger values of these parameters could be more suitable for noisy domains. Another crucial parameter is the *a* parameter used in the significance test we have employed. We have seen that in noise-free domains, $\alpha = 0.0$ will produce good results, while in noisy domains higher values $\alpha > 0.0$ must be used. We have also seen that the setting of this parameter is very sensitive: too low a setting may lead to exploding costs, while too high a setting may lead to over-generalization. Efficient methods for automating this choice would be highly desirable.

Another important question is how an extension of I-WIN that handles numeric data with thresholding will affect the performance of the algorithm. We expect that the fact that fewer thresholds have to be considered at lower example set sizes will have a positive effect on the run-time performance of windowing, but may have a negative effect on the accuracy of the learned rules. This hypothesis has been stated before [Catlett, 1992], but has never been empirically verified. In fact, we would not be surprised, if a lower set of potential thresholds, like the ones contained in the current window, gave the algorithm less chance for overfitting and could thus even increase predictive accuracy.

It lies in the nature of windowing that it can only work successfully, if there is some redundancy in the domain, i.e. that at least some of the rules of good theory can be learned from a subset of the given training examples. In [Ftirnkranz, 1997a] we present an example for a noise-free dataset, where this assumption does not hold, and consequently windowing is not effective. Techniques for estimating the redundancy of a domain would be another valuable point for further research.

7 Related Work

There have been several approaches that use subsampling algorithms that differ from windowing. For decision tree algorithms it has been proposed to use dynamic subsampling at each node in order to determine the optimal test. This idea

[2] Because of space limitations the graphs showing these results had to be omitted. They can be found in the technical report OEFAI-TR-97-07, which is available from www. ai . univie . ac. at.

[3] We discretized the domain's 7 continuous variables in a fairly arbitrary fashion. For example, we have mapped the age of the patient into 10 years intervals, as e.g. [1 ... 10], [11 ... 20), etc. The six other continuous attributes contain numerous missing values. For each of these attributes an additional binary attribute indicates whether the feature is present or not. We collapsed these pairs of attributes into single attributes, using a designated value as an indication that this attribute has not been measured, and 5 to 10 additional values that code the discretized measurements. We have also turned the problem into a binary problem, where the task is to discriminate the 2401 instances with a diagnosed condition from the 6771 instances with no such condition.

has been originally proposed, but not evaluated in [Breiman et al, 1984]. This approach was further explored in Catlett 's work on peepholing [Catlett, 1992], which is a sophisticated procedure for using subsampling to eliminate unpromising attributes and thresholds from consideration.

Most closely related to windowing is uncertainty sampling [Lewis and Catlett, 1994]. Here the new window is not selected on the basis of misclassified examples, but on the basis of the learner's confidence in the learned theory. The examples that are classified with the least confidence will be added to the training set in the next iteration.

A different approach that successively increases the current learning window is presented in [John and Langley, 1996]. Here examples are added until an extrapolation of the learning curve does no longer promise significant gains. However, the authors note that this technique can in general only gain efficiency for incremental learning algorithms.

Work on partitioning, i.e. splitting the example space into segments of equal size and combining the rules learned on each partition, has also produced promising results in noisy domains, but has substantially decreased learning accuracy in non-noisy domains [Domingos, 1996]. Besides, the technique seems to be tailored to a specific learning algorithm and not generally applicable.

8 Summary

We have presented a noise-tolerant version of windowing that is based on a separate-and-conquer strategy. Good rules that have been found at smaller sizes of the training window will be kept in the final theory, and all examples they cover will be removed from the training set, thus reducing the size of the window in the next iteration. Examples are added to the window by sampling from examples that are covered by insignificant rules or positive examples that are not covered by any rule of the previous iteration. Although we have used a fixed noise-tolerant rule learning algorithm throughout the paper, the presented windowing technique could use any noise-tolerant rule learner as its basic algorithm.

Acknowledgements

This research is sponsored by the Austrian Funds zur Forderung der Wissenschaftlichen Forschung (FWF). Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science and Transport. I would like to thank Ray Mooney for making his Common Lisp ML library publicly available, which has been used for the implementation of our programs; Gerhard Widmer for his comments on an earlier version of this paper; the maintainers of and contributors to the UCI machine learning repository; and the three anonymous reviewers for valuable suggestions and pointers to relevant literature.

References

(Breiman et al., 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth & Brooks, Pacific Grove, CA, 1984.

(Catlett, 1991] Jason Catlett. Megainduction: A test flight. In L.A. Birnbaum and G.C. Collins, editors, Proceedings of the Sth International Workshop on Machine Learning (ML-91), pages 596-599, Evanston, IL, 1991. Morgan Kaufmann.

(Catlett, 1992] Jason Catlett. Peepholing: Choosing attributes efficiently for megainduction. In Proceedings of the 9th International Conference on Machine Learning (ML-91), pages 49-54. Morgan Kaufmann, 1992.

(Clark and Niblett, 1989] Peter Clark and Tim Niblett. The CN2 induction algorithm. Machine Learning, 3(4):261-283,1989.

(Cohen, 1995 J William W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, Proceedings of the 12th International Conference on Machine Learning (ML-95), pages 115-123, Lake Tahoe, CA, 1995. Morgan Kaufmann.

(Domingos, 1996] Pedro Domingos. Efficient specific-to-general rule induction. In E. Simoudis and J. Han, editors, Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), pages 319-322. AAAI Press, 1996.

(FUrnkranz and Widmer, 1994] Johannes FUrnkranz and Gerhard Widmer. Incremental Reduced Error Pruning. In W. Cohen and H. Hirsh, editors. Proceedings of the 11th International Conference on Machine Learning (ML-94), pages 70-77, New Brunswick, NJ, 1994. Morgan Kaufmann.

(FUrnkranz, 1997a] Johannes FUrnkranz. More efficient windowing. In Proceedings of the 14th National Conference on Artificial Intelligence (AAA1-97), Providence, RI, 1997. AAAI Press.

(FUrnkranz, 1997b] Johannes FUrnkranz. Separate-and-conquer rule learning. Artificial Intelligence Review, 1997. To appear.

(John and Langley, 1996] George H. John and Pat Langley. Static versus dynamic sampling for data mining. In E. Simoudis and J. Han, editors, Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), pages 367-370. AAAI Press, 1996.

(Kivinen and Mannila, 1994] Jyrki Kivinen and Heikki Mannila. The power of sampling in knowledge discovery. In Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-94), pages 77-85, 1994.

(Lewis and Cadett, 1994] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In Proceedings of the 11th International Conference on Machine Learning (ML-94), pages 148-156, New Brunswick, NJ, 1994. Morgan Kaufmann.

(Quinlan, i983] John Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, Machine Learning. An Artificial Intelligence Approach, pages 463-482. Tioga, Palo Alto, CA, 1983.

[Quinlan, 1993] John Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

(Toivonen, 1996] Hannu Toivonen. Sampling large databases for association rules. In Proceedings of the 22nd Conference on Very Large Data Bases (VLDB-96), pages 134-145, Mumbai, India, 1996.

[Wirth and Cadett, 1988] Jarryl Wirth and Jason Catlett. Experiments on the costs and benefits of windowing in ID3. In J. Laird, editor, Proceedings of the Sth International Conference on Machine Learning (ML-88), pages 87-99, Ann Arbor, MI, 1988. Morgan Kaufmann.

(Yang, 1996] Yiming Yang. Sampling strategies and learning efficiency in text categorization. In M. Hearst and H. Hirsh, editors, Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access, pages 88-95. AAAI Press, 1996. Technical Report SS-96-05.

LEARNING

Learning 4: Classification