Noise-Tolerant Windowing

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

OEFAI-TR-97-07

Abstract

Windowing has been proposed as a procedure for efficient memory use in the ID3 decision tree learning algorithm. However, previous work has shown that it may often lead to a decrease in performance, in particular in noisy domains. Following up on previous work, where we have shown that the ability of separate-and-conquer 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 developments in the areas of *Knowledge Discovery in Databases* (Fayyad, Piatetsky-Shapiro, & Smyth, 1996) and *Intelligent Information Retrieval* (Hearst & Hirsh, 1996) have again shown the limits of conventional machine learning algorithms. Dimensionality reduction using subsampling procedures has been recognized as a promising field of research (Lewis & Catlett, 1994; Yang, 1996).

A good deal of the lack of interest in windowing can be attributed to an empirical study (Wirth & Catlett, 1988) that showed that windowing is unlikely to gain any efficiency. The authors studied windowing with ID3 in various domains and concluded that windowing cannot be recommended as a procedure for improving efficiency. The best results were achieved in noise-free domains,

such as the *Mushroom* domain, where windowing was able to perform on the same level as ID3, while its performance in noisy domains was considerably worse.

In previous work (Fürnkranz, 1997), we have demonstrated that separate-and-conquer algorithms are better suited for windowing than divide-and-conquer algorithms, 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 (Fürnkranz, 1996). The basic learning algorithm we use, I-RIP, is based on I-REP (Fürnkranz & Widmer, 1994) and its successor RIPPER (Cohen, 1995). However, the algorithms presented in this paper do not depend on this choice; any other noise-tolerant rule learning algorithm could be used in I-RIP's place.

I-REP achieves noise-tolerance by first learning a single, complete and consistent rule on two thirds of the training data and then pruning this rule on the remaining third using a form of reduced error pruning (Quinlan, 1987). 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 to learn another rule until no more meaningful rules can be discovered. In (Cohen, 1995) it was shown that some of the parameters of the I-REP algorithm, like the pruning and stopping criteria, were not chosen optimally. We have implemented the I-REP algorithm as described in (Fürnkranz & Widmer, 1994), but using 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 as 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 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

In principle, windowing can be wrapped around any learning algorithm. It starts by picking a random sample of a user-settable size *InitSize* from the total set of *Examples* and uses it for learning a theory 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

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

Train = RANDOMSAMPLE(Examples,InitSize)
Test = Examples \ Train
repeat
    Theory = I-RIP(Train)
    NewTrain = ∅
    OldTest = ∅
    for Example ∈ Test
        Test = Test \ Example
        if CLASSIFY(Theory,Example) ≠ CLASS(Example)
            NewTrain = NewTrain ∪ Example
        else
            OldTest = OldTest ∪ Example
        if |NewTrain| = MaxIncSize
            exit for
    Test = APPEND(Test,OldTest)
    Train = Train ∪ NewTrain
until NewTrain = ∅
```

Figure 1: Extending I-RIP with windowing.

examples are tested and the next iteration starts with the new window. To make sure that all examples are tested in the first few iterations, our implementation appends the examples that have already been tested to the remaining examples in the test set, so that testing will start with new examples in the next iteration. We have named our implementation of a windowed version of I-RIP WIN-RIP (see figure 1).

4 The Problem of Noise in Windowing

An adaptation of the windowing technique described in the previous section 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 yield performance gains in noisy domains. The contrary is true: the main problem with windowing in noisy domains lies in the fact that windowing 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 the proportion of noisy examples in the learning window will 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 11,000 examples domain, where 10% of the examples are misclassified due to noise. In the next iteration, about 1000 misclassified noisy examples will be added to the window, thus doubling its size. 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 entirely random. It can be assumed that many more examples have to be added to the window in order to recover the

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

Train = RANDOMSAMPLE(Examples,InitSize)
Theory = ∅
repeat
    NewTheory = I-RIP(Train)
    for Rule ∈ NewTheory
        EVALUATERULE(Examples)
        NewTr = Train
        NewEx = Examples
        Candidates = ∅
        if SIGNIFICANT(Rule,Examples)
            Theory = Theory ∪ Rule
            NewTr = NewTr \ COVER(Rule,Train)
            NewEx = NewEx \ COVER(Rule,Examples)
        else
            Candidates = Candidates ∪ COVER(Rule,Examples)
    for Example ∈ POSITIVE(Examples)
        if Example ∉ COVER (NewTheory,Examples)
            Candidates = Candidates ∪ Example
    Examples = NewEx
    Train = NewTr ∪ RANDOMSAMPLE(Examples,MaxIncSize)
until Candidates = ∅
```

Figure 2: A noise-tolerant version of windowing.

structure that is inherent in the data. This hypothesis is consistent with the results of (Wirth & Catlett, 1988) and (Catlett, 1991), where the sensitivity of windowing to noisy data sets has been shown empirically.

5 A Noise-Tolerant Version of Windowing

The windowing algorithm described in (Fürnkranz, 1997), 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. Although these rules are consistent and will not add examples to the current window, they 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 (Fürnkranz, 1997) achieves this by separating rules that have been complete and 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.

The I-WIN algorithm shown in figure 2 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 complete data set and computes some quality measure from this information (procedure SIGNIFICANT in figure 2).

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, more elaborate criteria must be used. We have experimented with a variety of criteria known from the literature (like e.g. CN2's likelihood ratio significance test (Clark & Niblett, 1989)), but found that they are insufficient for our purposes. Eventually, we have settled to use 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. that the estimate derived from the complete training set is 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 complete training set.

The parameter α determines the degree to which the estimates $AccWin(r)$ and $AccTot(r)$ have to correspond. A setting of $\alpha = 0.0$ 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 complete 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. A typical setting in a noisy domain would be around $\alpha = 1.0$, but the parameter seems to be quite sensible. $\alpha = \infty$ 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, 1997) with the difference that WIN-DOS-95 only tests examples until it has collected $MaxIncSize$ new examples to add to

the current window and that it has to test the good rules learned in previous iterations in all subsequent iterations. I-WIN, on the other hand, tests a rule on the complete 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 complete training set. If it finds the rule to be insignificant, all examples that it covers and that are not already contained in the current window, will be considered as candidates that can be added to the window before the next iteration. Positive examples that have not been covered by any of the rules will also be considered as such candidates. From these candidate examples, $MaxIncSize$ will be randomly selected and added to the window before the next iteration. 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 last section. However, we stick to adding uncovered *positive* examples only, as after more and more rule have been discovered, the proportion of positive examples in the remaining training set will considerably decrease, so that the chances to pick 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 algorithms is unable to learn any rules from the current window. In that case, the algorithm in figure 2 will add $MaxIncSize$ uncovered positive examples to the current window. In our implementation of the algorithm we deal 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 do not cover any training examples that are not covered by other rules. This post-processing phase is described in more detail in (Fürnkranz, 1997).

6 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 resulted from different internal random splits of the training data by the I-RIP algorithm. 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

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

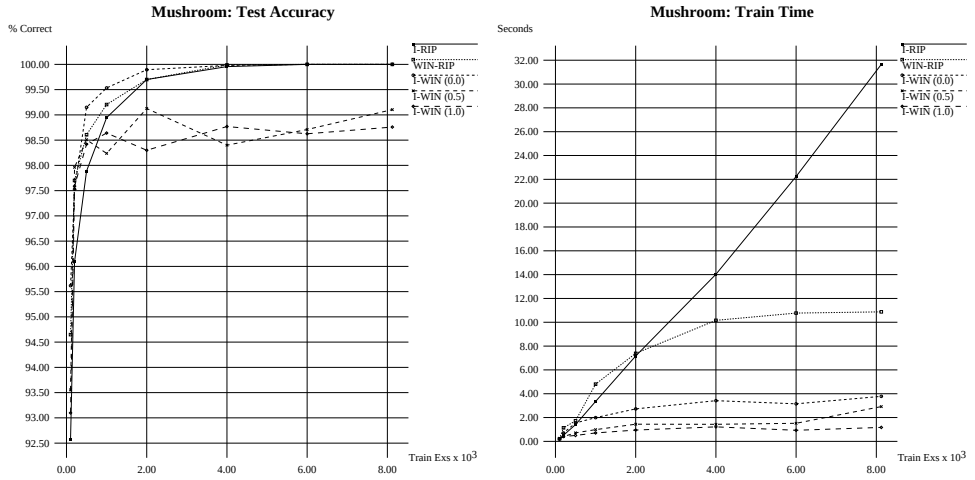

Figure 3: Results in the *Mushroom* domain.

found to perform well on noise-free domains (Fürnkranz, 1997). We have not yet made an attempt to evaluate their appropriateness for noisy domains.

First we have evaluated our 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, in (Wirth & Catlett, 1988) windowing with the decision tree learner ID3 could not achieve significant run-time gains over pure ID3, 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)).

Figure 3 shows the accuracy and run-time results for I-RIP, WIN-RIP, and three versions of I-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, although it is quite obvious that the three versions of I-WIN are the fastest. In terms of accuracy, no significant differences can be observed between I-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 (Fürnkranz & Widmer, 1994). I-WIN with $\alpha = 0.5$ and $\alpha = 1.0$ 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. The curves shown in figure 3 are quite similar to those of (Fürnkranz, 1997), where we have compared the results of a similar windowing algorithm to previous results on windowing with decision tree learning algorithms, which were less impressive, even when compared to regular windowing.

For testing the algorithms' noise-handling capabilities we have performed a series of experiments in a propositional version of the well-known KRK classi-

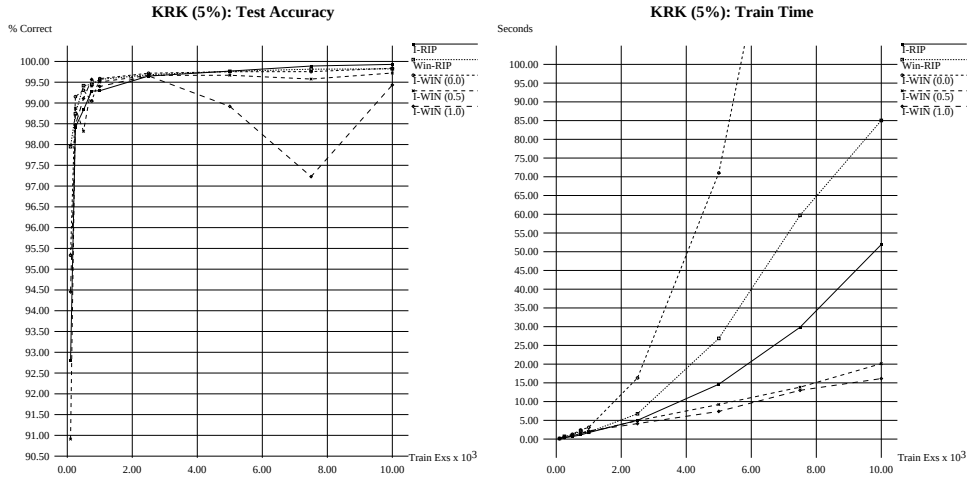

Figure 4: Results for 5% noise level in the KRK domain.

fication task, which is commonly used as a benchmark for relational learning algorithms. 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. The target concept 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. We have generated 10,000 noise-free examples in this domain, which were always used for testing the accuracies of the learned theories. In the training sets, which 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). On noise-free data sets the results were quite similar to those in the *Mushroom* domain, and are not shown here.

What happens when a noise-free windowing procedure is used in a noisy domains is shown in figure 4. It shows the results in the KRK domain with a very moderate noise level (5%). Regular windowing is almost twice as expensive as I-RIP, while our windowing procedure used in the noise-free setting, is even more expensive (it needs more than 300 secs. 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. For higher levels of noise, higher values of α will produce good results, as can be seen from figure 5, which shows the results for I-RIP and I-WIN ($\alpha = 1.0$) with 20% noise in the data.

Figure 6 shows the results for noise-levels varying from 0% to 100% (totally random data). We show the curves for I-RIP and I-WIN ($\alpha = 1.0$) for three

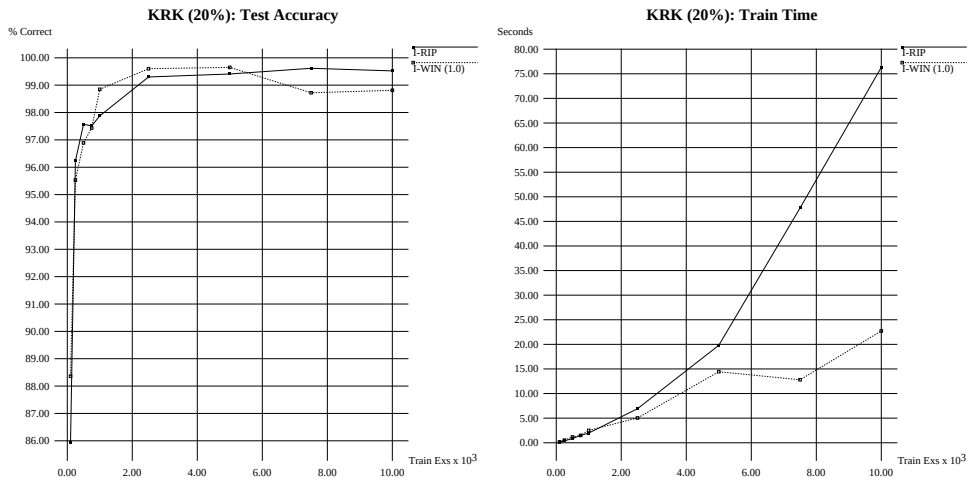

Figure 5: Results for 20% noise in the KRK domain.

different training set sizes (1000, 5000, 10000 examples). In terms of run-time, we see that the more random the data are, the less likely it is that the rules learned by I-RIP from the current window 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 complete data set.[2] However, for reasonable noise levels, which can be expected to occur in most real-world applications (say $\leq 30\%$), I-WIN significantly outperforms I-RIP at larger training set sizes (for 1000 examples no significant differences can be observed at the lower noise levels). The results in terms of run-time are very inconclusive with both algorithms having their ups and downs.

Currently, the implementation of our algorithms is limited to purely 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 simplified version of Quinlan's 9172 example thyroid diseases database. For this purpose 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,

[2] The results for I-WIN with noise-levels 75% and 100%, which have been omitted from the graph, have been 407 and 317 secs. respectively.

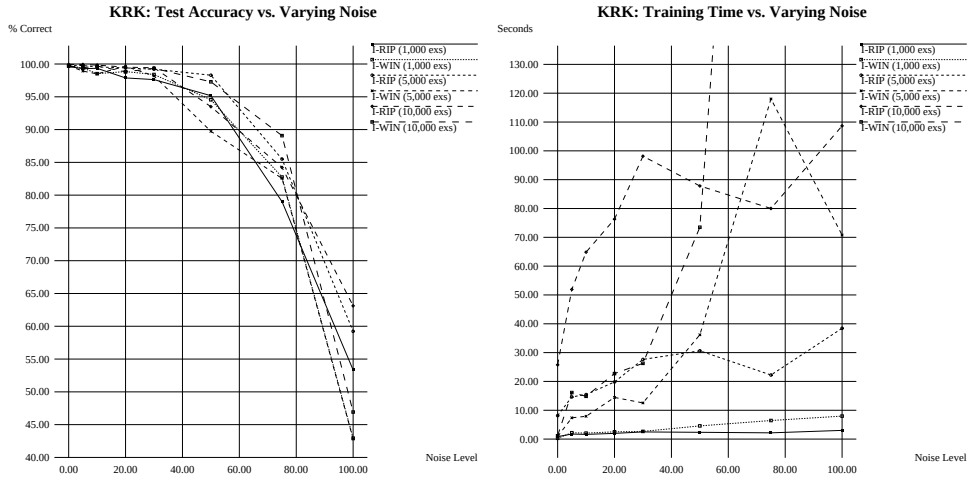

Figure 6: Results for varying noise levels and training set sizes in the KRK domain.

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. The default accuracy for this problem is 73.82%.

Figure 7 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 complete 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 rules learned at low window sizes on the complete training set.

7 Further Research

In its current form the implemented algorithms constitute only a first test-bed for an evaluation of the ideas presented in this paper. Several questions remain unanswered by this work and are subject to further research. Among them are

- 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 (Fürnkranz, 1997), but during the experiments reported in this paper, we have encountered some evidence that larger values of these parameters would be more suitable for noisy domains.

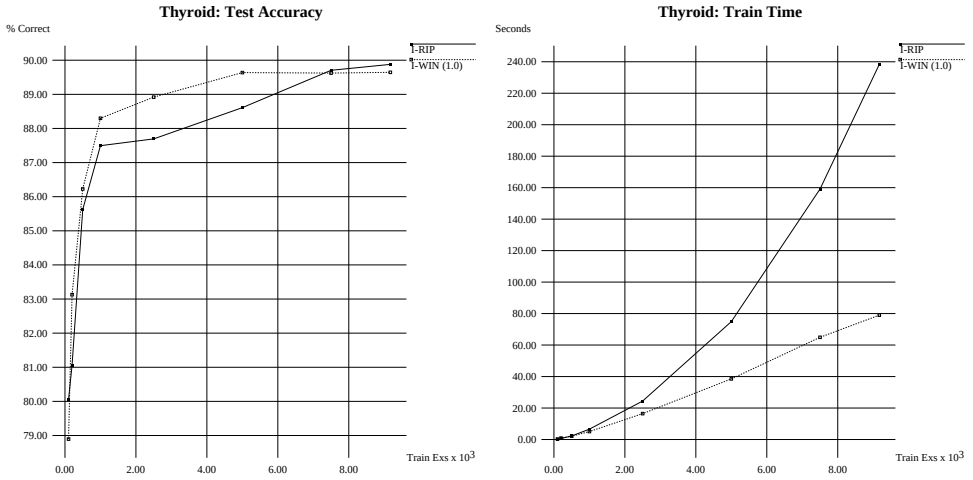

Figure 7: Results in the simplified *Thyroid* domain.

- Another problem is the α parameter used in the significance test we have employed. We have seen that in noise-free domains, it can be crucial that $\alpha = 0.0$, while in noisy domains higher values $\alpha > 0.0$ must be used. We have also seen that the setting of this parameter is very sensible: 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.

- How will the inclusion of an algorithm for handling numeric data with thresholding 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.

- The algorithms in this paper assume that the domains contain a reasonable level of redundancy, i.e. that at least some of the rules of good theory can be learned from a subset of the given training examples. In (Fürnkranz, 1997) 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.

- In (Fürnkranz, 1997) we have used an implementation-dependent measure for evaluating the complexity of the learning algorithms, namely the number of examples processed by the basic learning algorithm. This measure was highly correlated to run-time. However, this measure proved to be inadequate for the algorithms discussed in this paper. One of the reasons for this is our handling of iterations, where I-RIP returns the empty theory (end of section 5). Many runs end with a sequence of window-size

doubling events, which all return empty theories. These sequences highly affect the number-of-processed-examples measure, but only have a minor impact on run-time.

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 *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number P10489-MAT. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science, Transport, and the Arts. I would like to thank Ray Mooney for making his Common Lisp ML library publicly available, which has been used for the implementation of the programs, Gerhard Widmer, Johann Petrak, and Arthur Flexer for interesting discussions on several aspects of the paper, and the maintainers and contributers of the UCI machine learning repository.

References

Catlett, J. (1991). Megainduction: A test flight. In Birnbaum, L., & Collins, G. (Eds.), *Proceedings of the 8th International Workshop on Machine Learning (ML-91)*, pp. 596–599 Evanston, IL. Morgan Kaufmann.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*(4), 261–283.

Cohen, W. W. (1995). Fast effective rule induction. In Prieditis, A., & Russell, S. (Eds.), *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pp. 115–123 Lake Tahoe, CA. Morgan Kaufmann.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, *17*(3), 37–54.

Fürnkranz, J. (1996). Separate-and-conquer rule learning. Tech. rep. OEFAI-TR-96-25, Austrian Research Institute for Artificial Intelligence.

Fürnkranz, J. (1997). More efficient windowing. Tech. rep. OEFAI-TR-97-01, Austrian Research Institute for Artificial Intelligence.

Fürnkranz, J., & Widmer, G. (1994). Incremental Reduced Error Pruning. In Cohen, W., & Hirsh, H. (Eds.), *Proceedings of the 11th International Conference on Machine Learning*, pp. 70–77 New Brunswick, NJ. Morgan Kaufmann.

Hearst, M. A., & Hirsh, H. (Eds.). (1996). *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*. AAAI Press. Technical Report SS-96-05.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*. Morgan Kaufmann.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine Learning. An Artificial Intelligence Approach*, pp. 463–482. Tioga, Palo Alto, CA.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27*, 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Wirth, J., & Catlett, J. (1988). Experiments on the costs and benefits of windowing in ID3. In Laird, J. (Ed.), *Proceedings of the 5th International Conference on Machine Learning (ML-88)*, pp. 87–99 Ann Arbor, MI. Morgan Kaufmann.

Yang, Y. (1996). Sampling strategies and learning efficiency in text categorization. In Hearst, M., & Hirsh, H. (Eds.), *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*, pp. 88–95. AAAI Press. Technical Report SS-96-05.