Proceedings of the 2003 International Joint Conference on Neural Networks, 2764–2769.
Los Alamitos, CA: IEEE Press

Incremental Rule Learning with Partial Instance Memory for Changing Concepts

Marcus A. Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057-1232
maloof@cs.georgetown.edu

***Abstract*— Learning concepts that change over time is important for a variety of applications in which an intelligent system must acquire and use a behavioral profile. Computer intrusion detection, calendar scheduling, and intelligent user interfaces are three examples. An interesting class of methods for learning such concepts consists of algorithms that maintain a portion of previously encountered examples. Since concepts change over time and these methods store selected examples, mechanisms must exist to identify and remove irrelevant examples of old concepts. In this paper, we describe an incremental rule learner with partial instance memory, called AQ11-PM+WAH, that uses Widmer and Kubat's heuristic to adjust dynamically the window over which it retains and forgets examples. We evaluated this learner using the STAGGER Concepts and made direct comparisons to AQ-PM and to AQ11-PM, similar learners with partial instance memory. Results suggest that the forgetting heuristic is not restricted to FLORA2, the learner for which it was originally designed. Overall, results from this study and others suggest learners with partial instance memory converge more quickly to changing target concepts than algorithms that learn solely from new examples.**

I. Introduction

An interesting class of incremental learning methods consists of algorithms that maintain past concept descriptions [1] or previously encountered examples [2], [3], [4]. Most if not all of these efforts have been directed at learning concepts that change or drift [5]. If such learners successfully identify and leverage the stored examples pertaining to the new target concept, then they should have an advantage over algorithms that learn solely from new examples. Concepts that change over time occur in applications in which an intelligent system must acquire a model of human behavior. These include intrusion detection systems [4], market-basket analysis [6], e-mail importance ranking [2], calendar scheduling [7], and intelligent user interfaces [8].

Researchers have investigated several methods of retaining and then discarding examples of old concepts. One is to remove examples after a fixed period of time [1], [4], [9]. Another is to determine this period dynamically in response to changing performance [3].

We have investigated learners that maintain examples that lie on rule boundaries [2], [4], [10]. When new examples arrive, the learner produces new rules and forgets those examples that no longer fall on the boundaries of the new rules. To learn concepts that change over time, we have combined this method with mechanisms that remove examples after a fixed period of time [4], [10]. Thus far, we have considered applications of this method to computer misuse detection [4], [10], [11], shape detection [4], and e-mail importance ranking [2].

Recently, we described AQ11-PM [10], an incremental rule learner with *partial instance memory* that stores and uses examples falling on the boundaries of rules. By also forgetting examples after a fixed period of time, we showed empirically using the STAGGER Concepts [5] that it achieved comparable or superior predictive accuracy while maintaining fewer examples, as compared to similar systems designed for this task [3], [4].

In this paper, we extend our previous work by augmenting AQ11-PM [10] with a heuristic that determines dynamically the period over which the learner retains examples [3]. We evaluated this new system, AQ11-PM+WAH, on the STAGGER Concepts [5] and found it competitive with or superior to other learners in terms of predictive accuracy and the number of examples maintained, an outcome consistent with our previous studies [4], [10].

In the sections that follow, we provide a considerable discussion of background material for readers unfamiliar with rule learning. In particular, we cover batch and incremental rule learning, partial instance memory, and heuristics for retaining and forgetting examples. We then describe an experiment and its outcome in which we evaluated AQ11-PM+WAH using the STAGGER Concepts [5]. We also make direct comparisons to three other learners. We conclude the paper with a discussion of the experimental results and of plans for future work.

II. Background

A. Rule Learning

Off-line or batch concept learning involves recovering the unknown function f from the set $S = \{(\vec{x}, y) : y = f(\vec{x})\}$. Each element of S is a training example, each dimension of $\vec{x}$, an attribute, and each component of $\vec{x}$, an attribute value. For concept learning, y takes values of a finite set, and without loss of generality, we will assume that $y \in \{+, -\}$, indicating positive and negative examples of the concept f. In practice, the number of examples in S is a small percentage of the number possible, so we can only approximate f as $\hat{f}$. Numerous representations exist for $\hat{f}$, such as rules (e.g., [12]), trees (e.g., [13]), probabilities (e.g., [14]), and weights between a collection of non-linear processing units (e.g., [15]).

Rules are quite simple, consisting of the familiar form: **if** <*condition*> **then** <*action*>. For our purposes, <*condition*> is a conjunction of simple tests, each returning true if a

given attribute takes certain values (e.g., [*color* = *red, blue*]). <*action*> assigns a class label to a decision variable. Notice that each training example is a maximally specific rule, with each condition testing for only one value. Using single rules, we can represent conjunctive concepts. Using multiple rules, we can represent disjunctive concepts.

Given a set of rules and an unknown instance (i.e., $\vec{x}$), a rule learner's performance element uses the unknown's attribute values to satisfy a rule's conditions and assigns the rule's class label to the decision variable, which in turn, is the prediction for the unknown. Rules carve out decision regions in the space of examples, leaving some of the space uncovered, so an instance may not satisfy a rule. If an application requires the assignment of a class label, then we can flexibly match rules by, say, selecting the rule with the most true tests.

Learning rules from examples is an instance of the set-covering problem [16], with the added restriction that covers of one set cannot intersect with points of another. An NP-hard problem, the AQ algorithm is a heuristic method for inducing rules from examples under this constraint [17]. Other rule learning methods include CN2 [18], IREP [19], and RIPPER [20].

The AQ algorithm begins by selecting a positive example and generalizing it without covering any negative example, thereby forming a rule. It then removes the positive examples the rule covers and repeats with the remaining positive examples until covering them all. The algorithm forms rules for the negative class in the same fashion.

As an illustration, consider learning the concept of a person who is able to vote. In the United States, men and women who are at least eighteen years of age can vote. Using two attributes to represent examples, *gender* $\in \{M, F\}$ and *age* $\in \{0, 1, \ldots, 120\}$, assume we have the following training examples: $\{\langle M, 54 : +\rangle, \langle F, 42 : +\rangle, \langle M, 22 : +\rangle, \langle F, 32 : +\rangle, \langle F, 11 : -\rangle, \langle M, 14 : -\rangle, \langle M, 8 : -\rangle, \langle F, 16 : -\rangle\}$.

If we pick the positive example $\langle M, 22 : +\rangle$, its equivalent rule is **if** [*gender* = *M*] & [*age* = 22] **then** [*decision* = +]. By adding a test for the value *F* to the condition for the *gender* attribute, we can generalize the rule without intersecting any negative example, yielding **if** [*gender* = *M, F*] & [*age* = 22] **then** [*decision* = +]. One way to generalize the condition for the *age* attribute is to extend its range up to 120, which also yields a rule that does not intersect a negative example: **if** [*gender* = *M, F*] & [*age* = 22...120] **then** [*decision* = +]. Another is to extend the range of the condition for *age* down to 17, producing **if** [*gender* = *M, F*] & [*age* = 17...120] **then** [*decision* = +]. Since the condition for *gender* tests for all of the attribute's values, we can remove it, giving the rule **if** [*age* = 17...120] **then** [*decision* = +]. Applying the same procedure to the negative examples produces the rule **if** [*age* = 0...21] **then** [*decision* = −].

Clearly, these rules overlap, and for instances with values of *age* between 17 and 21, the performance element could return either or no decision. (It is possible for AQ to produce disjoint rules.) However, had the examples $\langle F, 18 : +\rangle$ and $\langle M, 17 : -\rangle$ been in the training set, then AQ would have produced rules correctly representing the target concept. This illustrates the importance of examples near the boundaries of concepts, an issue to which we return in Section II-D.

The previous two rules were *discriminant* descriptions [21] since they consist only of those attributes and values necessary for discriminating between objects of the two classes. AQ can also generate *characteristic* descriptions that consist of all attributes and values present in the training set. Notice that rules are axis-parallel hyper-rectangles in the space of examples, and that characteristic rules form the tightest hyper-rectangle about a set of examples.

Over the years, researchers have built numerous systems based on the AQ algorithm. Most are batch learning systems for inducing propositional rules, but INDUCE learns first-order rules [22]. There are also on-line versions of the AQ algorithm, and in the next section, we describe AQ11 [23], which learns rules incrementally with no instance memory.

B. On-line and Incremental Rule Learning

On-line learning is similar to batch learning, discussed in Section II-A, except training examples are distributed over time: $S_t = \{(\vec{x}, y) : y = f(\vec{x})\}$, for $t = 1, \ldots, \infty$. Now, $\hat{f}_t$ approximates f. On-line learning is important for applications in which we cannot collect all pertinent training data before applying an algorithm, and these include e-mail sorting [24], calendar scheduling [25], intelligent user interfaces [8], image analysis [26], and computer intrusion detection [4].

If the algorithm discards $\hat{f}_{t-1}$ and generates $\hat{f}_t$ from S_i, for $i = 1, \ldots, t$, then it is *on-line batch* or *temporal batch* with *full instance memory*. If the algorithm modifies $\hat{f}_t$ using $\hat{f}_{t-1}$ and S_t, then it is *incremental* with *no instance memory*. The first method has the advantage of learning from all available training data, but can be computationally expensive. Incremental learning can be faster, but induced rules may not be consistent with previously discarded examples. There are, however, incremental learners with full instance memory (e.g., [27], [28], [29]).

AQ11 [23] is an extension of the AQ algorithm and incrementally learns rules from training data with no instance memory. To illustrate its operation, assume we start with the two examples $\langle M, 54 : +\rangle$ and $\langle F, 11 : -\rangle$. From these, the AQ algorithm produces the rules **if** [*gender* = *M*] & [*age* = 12...120] **then** [*decision* = +] and **if** [*gender* = *F*] & [*age* = 0...53] **then** [*decision* = −]. When $\langle F, 42 : +\rangle$ arrives as the next example, AQ11 adds the value *F* to the condition for *gender* of the positive rule so it covers the new example. It can then drop the condition since it involves all values of *gender*, yielding the rule **if** [*age* = 12...120] **then** [*decision* = +].

When $\langle M, 14 : -\rangle$ arrives, AQ11 generalizes the negative rule in the same manner, producing **if** [*age* = 0...53] **then** [*decision* = −]. However, notice that the positive rule covers this new negative example, but AQ11 can specialize the rule by changing [*age* = 12...120] to [*age* = 15...120], thereby restricting the range that *age* can take, giving **if** [*age* = 15...120] **then** [*decision* = +].

To generalize and specialize rules, AQ11 takes advantage of the AQ algorithm's ability to generate covers that do not intersect points of other sets. As discussed previously, training examples (i.e., points) are maximally specific rules (i.e., covers). Therefore, to generalize a positive rule, AQ11 creates a set consisting of the rule and the new uncovered positive example, and uses the AQ algorithm to compute a new cover *against* a set containing negative rules and negative training examples. The resulting cover will not intersect the negative set.

As mentioned previously, AQ11, like most incremental learners with no instance memory, is susceptible to *ordering effects*. For example, if a learner has the rule **if** [*age* = 15...120] **then** [*decision* = +] and receives the example $\langle M, 13 : + \rangle$, it may generalize the rule to **if** [*age* = 13...120] **then** [*decision* = +], which covers the example $\langle M, 14 : - \rangle$, encountered in the previous trial. Properly setting AQ11's parameters can reduce such problems, but these difficulties have prompted some researchers to consider on-line learners with full instance memory (e.g., [27], [28], [29]). These learners maintain all previously seen examples, which minimizes ordering effects at the expense of memory, but as we see in the next section, these learners do not fare well when concepts drift or change.

C. Concept Drift

A particularly interesting problem for on-line learning is when f, the target concept, changes over time (i.e., f_t), known as the problem of drifting concepts, changing concepts, time-varying concepts, or nonstationary environments. In this area, there has been formal [30], [31] and empirical work with synthetic [3], [4], [5], [10] and real data sets [7], [32].

When concepts drift, the on-line schemes discussed in the previous section may not fare well. Any learner will perform poorly when a concept first changes, but schemes that learn with full instance memory will have great difficulty converging to the new target concept. The learner will perform poorly until examples from the new concept overwhelm those from the old one. Indeed, research suggests that learners that modify only their concept descriptions converge more quickly and to a higher accuracy on these tasks [5], [10].

If a learner stores and uses all previous examples, then when concepts change, perhaps it should discard all examples but those in the new training set. If the new and old concepts are completely disjoint, then this is a fine policy. However, if the concepts overlap, then there will be old examples useful for learning the new concept, and leveraging these will speed convergence to the new concept. This has prompted some researchers to consider schemes that store some of the examples from the input stream, the topic of the next section.

D. Incremental Learning with Partial Instance Memory

After concepts change, a learner with full instance memory will not converge as quickly to the new target concept as a learner that modifies its concept descriptions. However, if the former learner could identify those past examples that are consistent with the new target concept, then it should converge more quickly than an incremental learner with no instance memory.

Researchers have investigated several schemes of selecting examples from the input stream. The IB2 algorithm [33], an instance-based learner, stores only those examples it misclassifies. Since most misclassifications occur at the boundaries of concepts, IB2 stores examples near this interface. The original motivation for this algorithm was not concept drift, so if used for this purpose, additional mechanisms will be required to remove outdated examples. A simple method is to store examples over a fixed window of time [4], [9]. Naturally, performance is dependent on selecting the correct size of the window, so some researchers have examined heuristics for dynamically sizing these windows [3].

Similarly, AQ11-PM has *partial instance memory*, but it selects examples from the boundaries of its rules [10]. An extension of the AQ11 algorithm [23], AQ11-PM uses its rules to select examples from the training set that lie on the boundaries, storing these so-called *extreme examples* in memory [4]. When new examples arrive, the algorithm combines them with those held in memory and applies the AQ11 algorithm to modify the current set of rules. As rules change, AQ11-PM can forget examples in partial memory that no longer enforce a boundary, an *implicit forgetting* process since there is no explicit criteria for removing an example. However, to track drifting concepts, *explicit forgetting* may be necessary to, say, remove examples after a specified period of time.

To find extreme examples, AQ11-PM uses the AQ11 algorithm to induce rules from examples. As a post-processing step, it produces characteristic descriptions, which form the tightest hyper-rectangle about the training examples. The algorithm then manipulates the rules to select examples from the corners, edges, or surfaces [4].

Consider again the example of learning who can vote. Given the two examples $\langle M, 54 : + \rangle$ and $\langle F, 11 : - \rangle$, AQ11-PM will learn the rules **if** [*gender* = *M*] & [*age* = 12...120] **then** [*decision* = +] and **if** [*gender* = *F*] & [*age* = 0...53] **then** [*decision* = −]. The corresponding characteristic rules are **if** [*gender* = *M*] & [*age* = 54] **then** [*decision* = +] and **if** [*gender* = *F*] & [*age* = 11] **then** [*decision* = −]. It is trivial to see that these match the training examples, which the algorithm stores in memory with the rules.

Suppose $\langle M, 32 : + \rangle$ arrives next. The positive rule covers the example, but AQ11-PM specializes the negative rule, producing **if** [*gender* = *F*] & [*age* = 0...31] **then** [*decision* = −]. The new positive characteristic rule is **if** [*gender* = *M*] & [*age* = 32, 54] **then** [*decision* = +], which matches the new example, so AQ11-PM includes it in partial memory.

When the example $\langle M, 22 : + \rangle$ arrives, the positive rule covers it, AQ11-PM specializes the negative rule, and the new characteristic positive rule is **if** [*gender* = *M*] & [*age* = 22, 32, 54] **then** [*decision* = +]. However, the rule that matches the examples on its corners is **if** [*gender* = *M*] & [*age* = 22, 54] **then** [*decision* = +]. This rule matches the new example and the example $\langle M, 54 : + \rangle$ held in memory. The learner includes these in memory, but discards the example $\langle M, 32 : + \rangle$, since

the modified characteristic rule no longer matches it. We give the full details of this selection algorithm elsewhere [4].

Incrementally learning with partial instance memory may lessen some ordering effects. Certain applications may require additional mechanisms to remove examples from partial memory when they become too old, to weight recently seen examples more than others, or to prevent the removal of extreme examples from memory that no longer fall on a rule boundary. This is generally true for domains in which concepts change, and in the next section, we survey heuristics for forgetting old examples.

E. Heuristics for Forgetting

When using a learner with partial instance memory to acquire concepts that change, there must be a mechanism to remove irrelevant examples of the old concept. In the previous section, we discussed how AQ11-PM removes examples when they no longer fall on rule boundaries. We have also examined forgetting examples when they fall outside of a fixed window of time [4], [10]. A disadvantage of this approach is it limited to concepts that change with known and fixed periodicity.

Widmer and Kubat [3] investigated a heuristic approach to size such a window dynamically. Their window adjustment heuristic (WAH) takes into account current performance, whether accuracy is decreasing, and the coverage of the current concept descriptions, as shown in Figure 1. The user must set three parameters that determine thresholds for acceptable coverage and accuracy, but provided that concepts do not change too frequently, this heuristic will, in principle, size a forgetting window irrespective of the periodicity of change.

The heuristic takes four actions: reduce the window's size by 20%, reduce it by one time unit, make no change, and increase it by one. When concepts change, signaled by low coverage or by poor and decreasing accuracy, the heuristic quickly decreases the size of the window by 20%. As the learner acquires new concepts, the heuristic makes no change to the window's size or increases it gradually. When concepts are stable, signaled by high coverage and acceptable accuracy, the heuristic gradually decreases the window's size. In the next section, we present experimental results for a version of AQ11-PM augmented with the WAH.

Window-Adjustment-Heuristic ()

lc: threshold for low coverage, user-defined
hc: threshold for high coverage, user-defined
p: threshold for acceptable accuracy, user-defined
N: examples covered by the positive concept description
S: number of conditions in the positive description
Acc: accuracy of current concept descriptions
w: window size

if $(N/S < lc \vee (Acc < p \wedge decreasing(Acc)))$
    $\Delta w = -0.2w$;
else if $(N/S > 2.0 \times hc \wedge Acc > p)$
    $\Delta w = -1.0$;
else if $(N/S > hc \wedge Acc > p)$
    $\Delta w = 0.0$;
else
    $\Delta w = 1.0$;
$w = w + \Delta w$;
end.

Fig. 1. Heuristic for dynamically sizing a window of time [3].

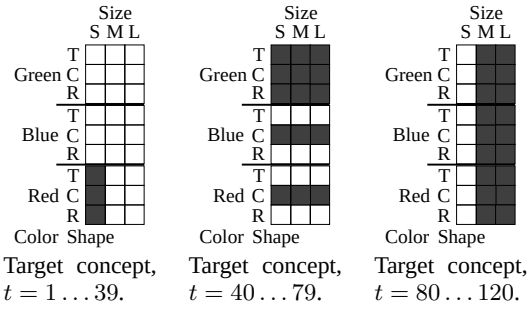

Fig. 2. Visualization of the STAGGER Concepts [4]. © 2000 Kluwer Academic Publishers.

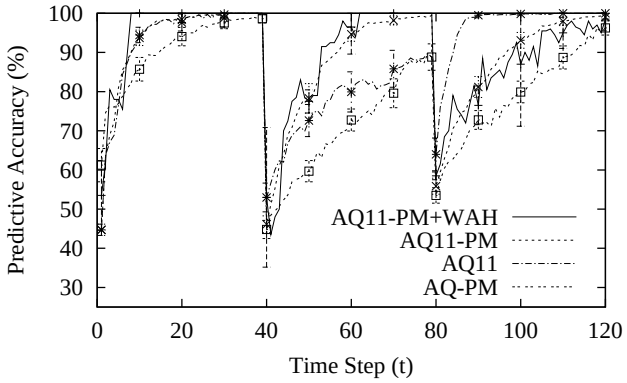

Fig. 3. Predictive accuracy for the STAGGER Concepts.

III. Experimental Results

In this section, we present experimental results for a version of AQ11-PM augmented with Widmer and Kubat's [3] window adjustment heuristic (WAH). We evaluated this new system,

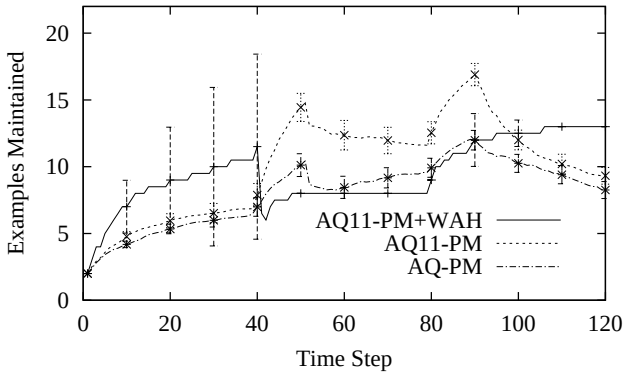

Fig. 4. Examples maintained in partial memory for the STAGGER Concepts.

called AQ11-PM+WAH, using the STAGGER Concepts [5], a benchmark for testing learners in the presence of concept drift. Using predictive accuracy and the number of examples maintained in memory as performance metrics, we made direct comparisons to AQ11 [23], AQ-PM [4], AQ11-PM [10]. (AQ-PM is a temporal-batch learner with partial instance memory.)

The STAGGER Problem [5] consists of three target concepts, as shown in Figure 2. Three attributes encode objects: size, taking values small, medium, and large; shape, taking values circle, triangle, and rectangle; and, color, taking values red, blue, and green. The presentation of examples occurs over three periods, each consisting of forty time steps. The target concept for the first period is [*size = small*] & [*color = red*]. For the second period, it is [*color = green*] $\vee$ [*shape = circle*]. For the third, the target is [*size = medium, large*]. For each of the 120 time steps, the learner receives either a positive or negative example of the target concept drawn randomly. After adjusting its concept descriptions, as a test, the learner must classify 100 examples, also drawn randomly.

To conduct the experiment, we repeated this protocol fifty times for the four learners, measuring predictive accuracy and the number of examples held in memory. The parameters for the WAH were $lc = 0$, $hc = 4$, and $p = 80$. In Figure 3, we present the accuracy averaged over these fifty runs with 95% confidence intervals.

On the first concept, AQ11-PM+WAH quickly achieved 100% accuracy, and on the second, it was competitive with AQ11-PM operating with a fixed window of fifty time units (i.e., the learner forgets examples older than fifty). However, on the third concept, AQ11-PM appears to perform slightly better without adaptive windowing. AQ11, despite its poor performance on the second concept, converged to the third concept more quickly than did the other learners, an issue to which we return in the next section. Nevertheless, AQ11-PM with and without the WAH outperformed AQ-PM on all three concepts.

Turning to memory requirements, in Figure 4, we see the average number of examples the partial-memory learners maintained over fifty runs, again with 95% confidence intervals. (Note that AQ11 does not maintain examples, which is why a curve for it is not present.) On average, AQ11-PM and AQ11-PM+WAH maintained roughly the same number of examples: 10.1 ± 0.8 and 9.5 ± 1.7, respectively. Although these learners maintained more examples than did AQ-PM, they also achieved higher predictive accuracy. What is striking about the performance of AQ11-PM+WAH is the relatively high variance during the acquisition of the first concept and the lack of variation during the second and third, an issue we discuss with others in the next section.

IV. Discussion

Previous results and those reported herein suggest that partial-memory learners, such as FLORA2 [3], AQ-PM [4], AQ11-PM [10], and AQ11-PM+WAH converge more quickly to new target concepts than do systems that learn incrementally by adjusting concept descriptions [5], [10]. The reason, we contend, is partial-memory learners can leverage examples in their store pertaining to the new target concept, which leads to faster convergence.

Results also indicate that AQ-PM [4], AQ11-PM [10], and AQ11-PM+WAH are competitive with FLORA2 [3] in terms of predictive accuracy, but the AQ systems maintain significantly fewer examples. FLORA2 uses the WAH to maintain a varying number of examples drawn consecutively from the input stream. As a result, its store may contain redundant and irrelevant examples. The AQ systems with partial memory store a nonconsecutive collection of examples that enforce rule boundaries. Consequently, the AQ systems maintain fewer examples and achieve comparable accuracy.

Our results are notable with one exception: AQ11 outperformed AQ11-PM and AQ11-PM+WAH on the third concept. (See Figure 3.) We have argued elsewhere [10] that AQ11 performed well on the third concept because it performed so poorly on the second. Our reasoning was that rules poorly representing the second concept would be easy to change when learning the third concept. Had AQ11 learned the second concept as well as AQ11-PM had, then its performance would have been equal to or worse than AQ11-PM (and AQ11-PM+WAH).

Further experimentation and analysis has not supported this hypothesis. To investigate, we trained AQ11 and AQ11-PM on each target concept until both attained 100% accuracy. We then switched to the next target concept. The plots of predictive accuracy from this experiment were similar to those presented in Figure 3, an outcome that does not support our hypothesis.

We gathered further evidence by plotting only the runs in which AQ11 achieved 100% on the second target concept. If our hypothesis is correct, then for these runs, AQ11 should perform worse than AQ11-PM on the third concept, but again, this was not the case. Presently, we are unsure why AQ11 consistently outperforms AQ11-PM on the third target concept. The implementations of these learners are slightly different, so we plan to examine the effects of these variations.

Naturally, the behavior of all of these systems, like most learners, is governed by a set of parameters. Determining such parameters for the AQ systems was a relatively easy exercise, but we were surprised at how sensitive the WAH was to its parameters. For instance, slight changes to the threshold for high coverage (*hc*) led to poor performance on the second concept. We determined these parameters manually, so it is possible that we could have eventually found parameters such that AQ11-PM+WAH outperforms AQ11 on the third STAGGER concept. An arduous task, we have begun investigating more principled ways of setting these parameters, such as greedy search and reinforcement learning [34]. Indeed, as we have shown elsewhere, learning algorithms can outperform carefully handcrafted heuristics [35]. Finally, since AQ11-PM does not necessarily maintain a consecutive sequence of examples, a different type of WAH may also be in order.

V. Conclusion

In this paper, we evaluated AQ11-PM+WAH, an incremental rule learner with partial instance memory that uses a heuristic to adjust dynamically the window over which it retains examples. We showed empirically using the STAGGER Concepts that the new method performed as well as or better than similar methods designed for this task. Our method is one of a class of algorithms that store a portion of the examples from the input stream. Researchers have shown that these methods are well-suited for domains in which concepts change or drift, such as computer security, calendar scheduling, and intelligent user interfaces. We hope that this and future work will lead to more general and robust learning algorithms for these applications.

Acknowledgment

This research was conducted in the Department of Computer Science at Georgetown University. The author thanks Matthew Merzbacher for suggesting experiments for assessing differences in performance between AQ11 and AQ11-PM, and Tom Torsney-Weir and the anonymous reviewers for helpful comments on earlier drafts of the paper.

References

[1] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM Press, 2001, pp. 97–106.

[2] M. Maloof, "Progressive partial memory learning," Ph.D. dissertation, School of Information Technology and Engineering, George Mason University, Fairfax, VA, 1996.

[3] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, pp. 69–101, 1996.

[4] M. Maloof and R. Michalski, "Selecting examples for partial memory learning," *Machine Learning*, vol. 41, pp. 27–52, 2000.

[5] J. Schlimmer and R. Granger, "Beyond incremental processing: Tracking concept drift," in *Proceedings of the Fifth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press, 1986, pp. 502–507.

[6] G. Gupta, "Modeling customer dynamics using motion estimation in a value-based cluster space for large retail data-sets," Master's thesis, Department of Electrical and Computer Engineering, University of Texas, Austin, 2000.

[7] A. Blum, "Empirical support for Winnow and Weighted-Majority algorithms: Results on a calendar scheduling domain," *Machine Learning*, vol. 26, pp. 5–23, 1997.

[8] M. Maybury and W. Wahlster, Eds., *Readings in intelligent user interfaces*. San Francisco, CA: Morgan Kaufmann, 1998.

[9] G. Widmer, "Tracking context changes through meta-learning," *Machine Learning*, vol. 27, pp. 259–286, 1997.

[10] M. Maloof and R. Michalski, "Incremental learning with partial instance memory," in *Foundations of intelligent systems*, ser. Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, vol. 2366, pp. 16–27.

[11] ——, "A method for partial-memory incremental learning and its application to computer intrusion detection," in *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*. Los Alamitos, CA: IEEE Press, 1995, pp. 392–397.

[12] J. Fürnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, 1999.

[13] J. Quinlan, *C4.5: Programs for machine learning*. San Francisco, CA: Morgan Kaufmann, 1993.

[14] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of the Tenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press, 1992, pp. 223–228.

[15] C. Bishop, *Neural networks for pattern recognition*. Oxford: Clarendon Press, 1995.

[16] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[17] R. Michalski, "On the quasi-minimal solution of the general covering problem," in *Proceedings of the Fifth International Symposium on Information Processing*, vol. A3, 1969, pp. 125–128.

[18] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, pp. 261–284, 1989.

[19] J. Fürnkranz and G. Widmer, "Incremental reduced error pruning," in *Proceedings of the Eleventh International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1994, pp. 70–77.

[20] W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1995, pp. 115–123.

[21] R. Michalski, "Pattern recognition as rule-guided inductive inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 4, pp. 349–361, 1980.

[22] ——, "A theory and methodology of inductive learning," in *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonell, and T. Mitchell, Eds. San Francisco, CA: Morgan Kaufmann, 1983, vol. 1, pp. 83–134.

[23] R. Michalski and J. Larson, "Incremental generation of VL_1 hypotheses: The underlying methodology and the description of program AQ11," Department of Computer Science, University of Illinois, Urbana, Technical Report UIUCDCS-F-83-905, 1983.

[24] W. Cohen, "Learning rules that classify e-mail," in *Machine learning in information access: Papers from the 1996 AAAI Spring Symposium*. Menlo Park, CA: AAAI Press, 1996, pp. 18–25, Technical Report SS-96-05.

[25] T. Mitchell, "Machine learning and data mining," *Communications of the ACM*, vol. 42, no. 11, pp. 30–36, Nov. 1999.

[26] M. Maloof, "An initial study of an adaptive hierarchical vision system," in *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 2000, pp. 567–573.

[27] R. Reinke and R. Michalski, "Incremental learning of concept descriptions: A method and experimental results," in *Machine Intelligence 11*, J. Hayes, D. Michie, and J. Richards, Eds. Oxford: Clarendon Press, 1988, pp. 263–288.

[28] P. Utgoff, "ID5: An incremental ID3," in *Proceedings of the Fifth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1988, pp. 107–120.

[29] P. Utgoff, N. Berkman, and J. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, pp. 5–44, 1997.

[30] A. Kuh, T. Petsche, and R. Rivest, "Learning time-varying concepts," in *Advances in Neural Information Processing Systems 3*. San Francisco, CA: Morgan Kaufmann, 1991, vol. 3, pp. 183–189.

[31] J. Case, S. Jain, S. Kaufmann, and A. Sharma, "Predictive learning models for concept drift," *Theoretical Computer Science*, vol. 268, no. 2, pp. 323–349, 2001.

[32] M. Black and R. Hickey, "Classification of customer call data in the presence of concept drift and noise," in *Soft-Ware 2002: Computing in an Imperfect World*, ser. Lecture Notes in Computer Science, 2002, vol. 2311, pp. 74–87.

[33] D. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[34] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[35] M. Maloof, P. Langley, T. Binford, and R. Nevatia, "Generalizing over aspect and location for rooftop detection," in *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision (WACV '98)*. Los Alamitos, CA: IEEE Press, 1998, pp. 194–199.