Proceedings of the Third International IEEE Conference on Data Mining, 123–130.
Los Alamitos, CA: IEEE Press

Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift*

Jeremy Z. Kolter and Marcus A. Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057-1232, USA
{jzk, maloof}@cs.georgetown.edu

Abstract

Algorithms for tracking concept drift are important for many applications. We present a general method based on the Weighted Majority algorithm for using any on-line learner for concept drift. Dynamic Weighted Majority (DWM) maintains an ensemble of base learners, predicts using a weighted-majority vote of these "experts", and dynamically creates and deletes experts in response to changes in performance. We empirically evaluated two experimental systems based on the method using incremental naive Bayes and Incremental Tree Inducer (ITI) as experts. For the sake of comparison, we also included Blum's implementation of Weighted Majority. On the STAGGER Concepts and on the SEA Concepts, results suggest that the ensemble method learns drifting concepts almost as well as the base algorithms learn each concept individually. Indeed, we report the best overall results for these problems to date.

1. Introduction

Learning algorithms that track *concept drift* [23] are important for many domains. Such algorithms must be applicable to a variety of problems, both large and small. They must be robust to noise. Finally, they must converge quickly to target concepts with high accuracy.

In this paper, we present an *ensemble method* that uses on-line learning algorithms to track drifting concepts. It is based on the Weighted Majority algorithm [15], but we added mechanisms to create and delete experts dynamically in response to changes in performance. Hence, we call this new method *Dynamic Weighted Majority* (DWM).

Using an incremental version of naive Bayes and Incremental Tree Inducer [25] as base algorithms, we evaluated the method using two synthetic problems involving concept drift: the STAGGER Concepts [23] and the "SEA Concepts", a problem recently proposed in the data mining community [24]. For the sake of comparison, we also evaluated Blum's [3] implementation of Weighted Majority on the STAGGER Concepts. We did so because it is an obvious evaluation that to our knowledge has never been published. Our results suggest that DWM learns drifting concepts almost as well as the base algorithms learn each concept individually (i.e., with perfect forgetting).

We make three contributions. First, we present a general method for using any on-line learning algorithm for problems involving concept drift. Second, we conducted a thorough empirical study of the method, which included two incremental learners as base algorithms, two synthetic problems that have appeared in the literature, and five other methods for the sake of comparison. Third, because of our comprehensive evaluation, we firmly place our results in context with those reported previously. To the best of our knowledge, we present the best overall results for these problems to date.

2. Background and Related Work

The on-line learning task is to acquire a set of concept descriptions from labeled training data distributed over time. This type of learning is important for many applications, such as computer security, intelligent user interfaces, and market-basket analysis. An important class of problems for on-line learning involves target concepts that change over time [23]. For instance, customer preferences change as new products and services become available. Algorithms for coping with concept drift must converge quickly and accurately to new target concepts, while being efficient in time and space.

Researchers have proposed and evaluated several algorithms for coping with concept drift. STAGGER [23] was the first designed expressly for concept drift, as were many of the algorithms that followed, such as FLORA2 [26], AQ-PM [18], AQ11-PM [19], and AQ11-PM-WAH [17].

Although researchers have not yet established the degree to which these algorithms scale to large problems, some have been designed expressly for learning time-varying concepts from large data sets [11,24]. Other algorithms are amendable to such problems because of their formal properties, at least in theory [3, 14, 15]. One such algorithm in this category is Weighted Majority [15]. It is a method for weighting and combining the decisions of "experts", each of which is a learning method. For instance, Blum [3] used pairs and triples of features as experts, which returned the majority vote over the most recent k predictions.

The algorithm begins by creating a set of experts and assigning a weight to each. When a new instance arrives, the algorithm passes it to and receives a prediction from each expert. The algorithm predicts based on a weighted-majority vote of the expert predictions. If an expert incorrectly classifies the example, then the algorithm decreases its weight by a multiplicative constant. Winnow [14] is similar to Weighted Majority, except that experts may abstain, and thus have been called "specialists" [3].

Blum [3] evaluated variants of Weighted Majority and Winnow on a calendar-scheduling task and results suggested that the algorithms responded well to concept drift and executed fast enough to be useful for real-time applications. However, using pairs of features requires $\binom{n}{2}$ experts, where n is the number of relevant features (i.e., attribute-value pairs), which makes the direct application of these implementations impractical for most data mining problems. In one case, when learning the scheduling preferences of one user using 34 attributes, Weighted Majority and Winnow required 59,731 experts and specialists, respectively.

The advantage of Weighted Majority and Winnow is that they provide a general scheme for weighting any fixed collection of experts. However, since there are no mechanisms for dynamically adding or removing new experts or specialists, they are restricted to problems for which we can determine *a priori* the number required. We provide a remedy in the next section, and in Section 4.1, we show that by using more sophisticated base algorithms, we can reduce the number of experts.

Weighted Majority and Winnow are ensemble methods, and in an off-line setting, such methods create individual classifiers and combine the predictions of these classifiers into a single prediction. For example, bagging [4] involves sampling with replacement from a data set, building a classifier using each sample, and predicting the majority prediction of the individual classifiers. Boosting [9] likewise creates a series of classifiers, albeit with a different method, weighting each classifier based on its performance. Several empirical evaluations suggest that ensembles perform better than do single classifiers [1, 5, 9, 21].

More recently, there has been work on ensemble methods for on-line learning tasks [8] and for concept drift [24]. Unfortunately, in an on-line setting, it is less clear how to apply ensemble methods directly. For instance, with bagging, when one new example arrives that is misclassified, it is too inefficient to resample the available data and learn new classifiers. One solution is to rely on the user to specify the number of examples from the input stream for each base learner [7], but this approach assumes we know a great deal about the structure of the data stream and is likely to be impractical for drifting concepts. There are on-line boosting algorithms that reweight classifiers [8], but these assume a fixed number of classifiers. Again, this could be a strong assumption when concepts change, but we are unaware of any on-line boosting approaches that have been applied to the problem of concept drift.

The Streaming Ensemble Algorithm (SEA) [24] copes with concept drift with an ensemble of C4.5 classifiers [22]. SEA reads a fixed amount of data and uses it to create a new classifier. If this new classifier improves the performance of the ensemble, then it is added. However, if the ensemble contains the maximum number of classifiers, then the algorithm replaces a poorly performing classifier with the new classifier. Performance is measured over the most recent predictions and is based on the performance of both the ensemble and the new classifier.

Unfortunately, there are problems with this approach. One is that members of the ensemble stop learning after being formed. This implies that a fixed period of time will be sufficient for learning all target concepts. In addition, if concepts drift during this fixed period of time, the learner may not be able to acquire the new target concepts. Finally, replacing the worst performing classifier in an unweighted ensemble may not yield the fastest convergence to new target concepts. In the next section, we describe a new ensemble method that copes with concept drift and addresses these problems.

3. DWM: A New Ensemble Method for Concept Drift

Dynamic Weighted Majority (DWM), shown in Figure 1, maintains as its concept description an ensemble of learning algorithms, each referred to as an expert and each with an associated weight. Given an instance, the performance element polls the experts, each returning a prediction for the instance. Using these predictions and expert weights, DWM returns as the global prediction the class label with the highest accumulated weight.

The learning element, given a new training example, first

Dynamic-Weighted-Majority ($\{\vec{x}, y\}_n^1$)
$\{\vec{x}, y\}_n^1$: training data, feature vector and class label
β : factor for decreasing weights, $0 \le \beta < 1$
$c \in \mathbb{N}^*$: number of classes
$\{e, w\}_m^1$: set of experts and their weights
$\Lambda, \lambda \in \{1, \ldots, c\}$: global and local predictions
$\vec{\sigma} \in \mathbb{R}^c$: sum of weighted predictions for each class
θ: threshold for deleting experts
p: period between expert removal, creation, and weight update

```
for i = 1,...,n
    σ⃗ ← 0
    for j = 1,...,m
        λ = Classify(e_j, x⃗_i)
        if (λ ≠ y_i and i mod p = 0)
            w_j ← βw_j
        σ_λ ← σ_λ + w_j
    end;
    Λ = argmax_j σ_j
    if (i mod p = 0)
        w ← Normalize-Weights(w)
        {e, w} ← Delete-Experts({e, w}, θ)
        if (Λ ≠ y_i)
            m ← m + 1
            e_m ← Create-New-Expert()
            w_m ← 1
        end;
    end;
    for j = 1,...,m
        e_j ← Train(e_j, x⃗_i)
    output Λ
end;
end.
```

Figure 1. Algorithm for dynamic weighted majority (DWM).

polls each expert in the manner described previously. If an expert predicts incorrectly, then its weight is reduced by the multiplicative constant β.

DWM then determines the global prediction. If it is incorrect, then the algorithm creates a new expert with a weight of one. The algorithm normalizes expert weights by uniformly scaling them such that the highest weight will be equal to one. This prevents any newly added experts from dominating the decision making of existing ones. The algorithm also removes experts with weights less than the user-defined threshold θ. Finally, DWM passes the training example to each expert's learning element. Note that normalizing weights and incrementally training all experts gives the base learners an opportunity to recover from concept drift. Large and noisy problems required the parameter p, which governs the frequency that DWM creates experts, removes them, and updates their weights (i.e., reduction and normalization.)

We implemented DWM using two different base learners: an incremental version of naive Bayes and an incremental decision-tree learner. For symbolic attributes, our incremental version of naive Bayes (NB) stores as its concept description counts for the number of examples of each class and for each attribute value given the class. Learning, therefore, entails incrementing the appropriate counts given the new instance. During performance, the algorithm uses the stored counts to compute the prior probability of each class, $P(C_i)$, and the conditional probability of each attribute value given the class, $P(v_j|C_i)$. Then, under the assumption that attributes are conditionally independent, it uses Bayes' rule to predict the most probable class, C, given by

$$C = \operatorname*{argmax}_{C_i} P(C_i) \prod_j P(v_j|C_i).$$

For continuous attributes, our implementation stores for each class the sum of the attribute values and the sum of the squared values. Learning simply entails adding an attribute's value and the square of that value to the appropriate sum. During performance, the implementation uses the example count and the sums to compute the mean (μ) and variance (σ^2). Then, assuming that the jth attribute's values are normally distributed, it computes

$$P(v_j|C_i) = \Delta v_j \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(v_j-\mu)^2/2\sigma^2},$$

where Δv_j is the size of interval in which the random variable for the attribute lies. (See John and Langley [12] for details.) We will refer to the system with naive Bayes as DWM-NB.

The Incremental Tree Inducer (ITI) [25] is a complex algorithm, so we will be unable to describe it fully here. Briefly, ITI uses as its concept description a decision tree with only binary tests. In internal nodes, ITI stores frequency counts for symbolic attributes and a list of observed values for continuous attributes. In leaf nodes, it stores examples. ITI updates a tree by propagating a new example to a leaf node. During the descent, the algorithm updates the information at each node, and upon reaching a leaf node, determines if the tree should be extended by converting the leaf node to a decision node. A secondary process examines whether the tests at each node are most appropriate, and if not, restructures the tree accordingly. We will refer to the system with ITI as DWM-ITI.

4. Empirical Study and Results

In this section, we present experimental results for DWM-NB and DWM-ITI. We evaluated both systems on the STAGGER Concepts [23], a standard benchmark for evaluating how learners cope with drifting concepts. We also included Blum's implementation of Weighted Majority [3] for the sake of comparison. We know of no published results for this algorithm on the STAGGER Concepts. Finally, in an effort to determine how our method scales to larger problems involving concept drift, we also evaluated DWM-NB on the SEA Concepts [24], a problem recently proposed in the data mining community.

We did not include any UCI data sets [2] in our evaluation because naive Bayes, ITI, and ensemble methods in general, have been well studied on many of these tasks (e.g., [1,5,13,16,21,24,25]). Instead, we chose to evaluate the methods on problems involving concept drift, on which their performance is less understood.

4.1. The STAGGER Concepts

The STAGGER Concepts [23] comprise a standard benchmark for evaluating a learner's performance in the presence of concept drift. Each example consists of three attribute values: *color* $\in$ {*green, blue, red*}, *shape* $\in$ {*triangle, circle, rectangle*}, and *size* $\in$ {*small, medium, large*}. The presentation of training examples lasts for 120 time steps, and at each time step, the learner receives one example. For the first 40 time steps, the target concept is $color = red \wedge size = small$. During the next 40 time steps, the target concept is $color = green \vee shape = circle$. Finally, during the last 40 time steps, the target concept is $size = medium \vee size = large$.

To evaluate the learner, at each time step, one randomly generates 100 examples of the current target concept, presents these to the performance element, and computes the percent correctly predicted. In our experiments, we repeated this procedure 50 times and averaged the accuracies over these runs. We also computed 95% confidence intervals.

We evaluated DWM-NB, DWM-ITI, and Blum's Weighted Majority [3] with pairs of features as experts on the STAGGER Concepts. All of the Weighted Majority algorithms halved an expert's weight when it made a mistake (i.e., $\beta = 0.5$). For Blum's Weighted Majority, each expert maintained a history of only its last prediction (i.e., $k = 1$), under the assumption that this setting would provide the most reactivity to concept drift. Finally, for DWM, we set it to update its weights and create and remove experts every time step (i.e., $p = 1$). The algorithm removed experts when their weights fell below 0.01 (i.e., $\theta = 0.01$). Pilot studies indicated that these were the optimal settings for p and k; Varying β affected performance little; The selected value for θ did not affect accuracy, but did reduce the number of experts considerably.

For the sake of comparison, in addition to these algorithms, we also evaluated naive Bayes, ITI, naive Bayes with perfect forgetting, and ITI with perfect forgetting. The "standard" or "traditional" implementations of naive Bayes and ITI provided a worst-case evaluation, since these systems have not been designed to cope with concept drift and learn from all examples in the stream regardless of the target concept. The implementations with perfect forgetting, which is the same as training the methods on each target concept individually, provided a best-case evaluation, since the systems were never burdened with examples or concept descriptions from previous target concepts.

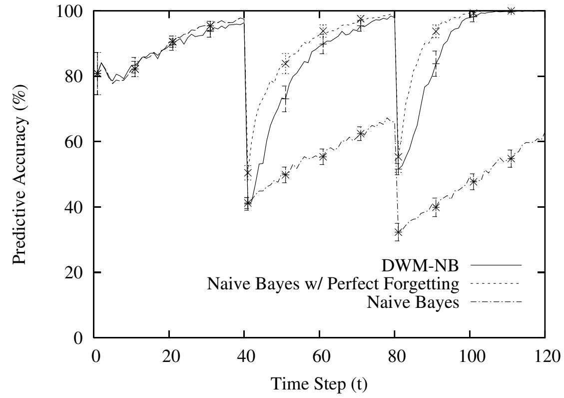

Figure 2. Predictive accuracy for DWM-NB on the STAGGER Concepts.

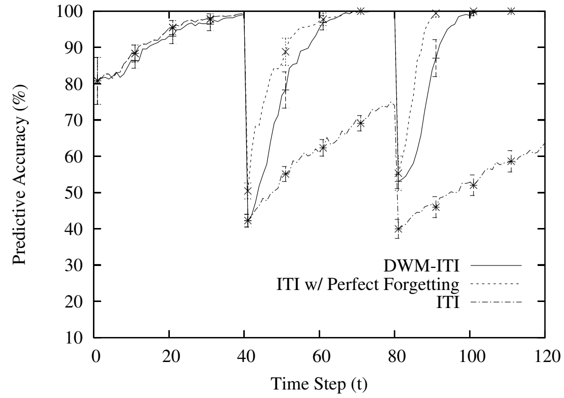

Figure 3. Predictive accuracy for DWM-ITI on the STAGGER Concepts.

Figure 2 shows the results for DWM-NB on the STAGGER Concepts. As expected, naive Bayes with perfect forgetting performed the best on all three concepts, while naive Bayes without forgetting performed the worst. DWM-NB performed almost as well as naive Bayes with perfect forgetting, which converged more quickly to the target concept. Nonetheless, by time step 40 for all three target concepts, DWM-NB performed almost as well as naive Bayes with perfect forgetting. (We place these results in context with related work in the next section.)

DWM-ITI performed similarly, as shown in Figure 3,

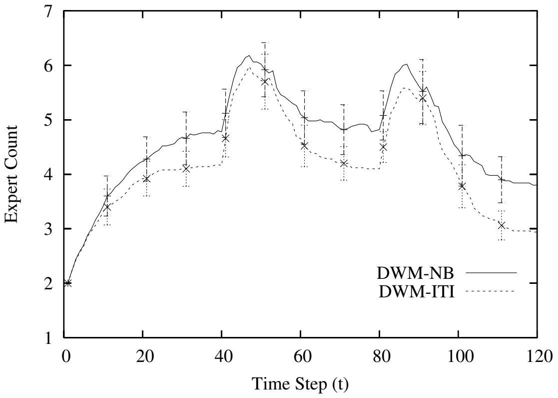

Figure 4. Number of experts maintained for DWM-NB and DWM-ITI on the STAGGER Concepts.

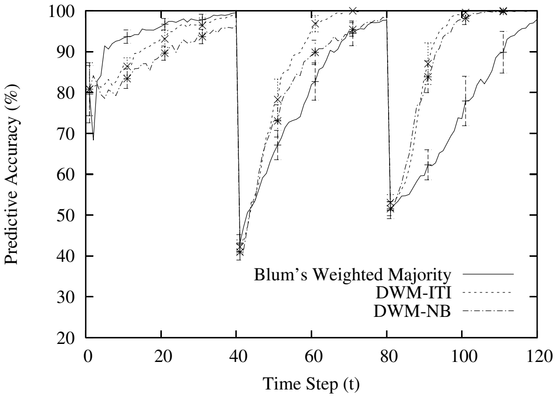

Figure 5. Predictive accuracy for DWM-ITI, DWM-NB, and Blum's Weighted Majority on the STAGGER Concepts.

achieving accuracies nearly as high as ITI with perfect forgetting. DWM-ITI converged more quickly than did DWM-NB to the second and third target concepts, but if we compare the plots for naive Bayes and ITI with perfect forgetting, we see that ITI converged more quickly to these target concepts than did naive Bayes. Thus, the faster convergence is due to differences in the base learners rather than to something inherent to DWM.

In Figure 4, we present the average number of experts each system maintained over the fifty runs. On average, DWM-ITI maintained fewer experts than did DWM-NB, and we attribute this to the fact that ITI performed better on the individual concepts than did naive Bayes. Since naive Bayes made more mistakes than did ITI, DWM-NB created more experts than did DWM-ITI. We can also see in the figure that the rates of removing experts is roughly the same for both learners.

Finally, Figure 5 shows the results from the experiment involving Blum's implementation of Weighted Majority [3]. This learner outperformed DWM-NB and DWM-ITI on the first target concept, performed comparably on the second, and performed worse on the third. We evaluated Blum's implementation of Weighted Majority that used pairs of features as experts. The STAGGER Concepts consist of three attributes, each taking one of three possible values. Therefore, this implementation of Weighted Majority maintained $\binom{9}{2} = 36$ experts throughout the presentation of examples, as compared to the maximum of six that DWM-NB maintained. Granted, pairs of features are much simpler than the decision trees that ITI produced, but our implementation of naive Bayes was quite efficient, maintaining twenty integers for each expert. There were occasions when Weighted Majority used less memory than did DWM-NB, but we anticipate that using more sophisticated classifiers, such as naive Bayes, instead of pairs of features, will lead to scalable algorithms, which is the topic of the next section.

4.2. Performance on a Large Data Set with Drift

To determine how well DWM-NB performs on larger problems involving concept drift, we evaluated it using a synthetic problem recently proposed in the data mining community [24]. This problem, which we call the "SEA Concepts", consists of three attributes, $x_i \in \mathbb{R}$ such that $0.0 \leq x_i \leq 10.0$. The target concept is $x_1 + x_2 \leq b$, where $b \in \{7, 8, 9, 9.5\}$. Thus, x_3 is an irrelevant attribute.

The presentation of training examples lasts for 50,000 time steps. For the first fourth (i.e., 12,500 time steps), the target concept is with $b = 8$. For the second, $b = 9$; the third, $b = 7$; and the fourth, $b = 9.5$. For each of these four periods, we randomly generated a training set consisting of 12,500 examples. In one experimental condition, we added 10% class noise; in another, we did not. We also randomly generated 2,500 examples for testing. At each time step, we presented each method with one example, tested the resulting concept descriptions using the examples in the test set, and computed the percent correct. We repeated this procedure ten times, averaging accuracy over these runs. We also computed 95% confidence intervals.

On this problem, we evaluated DWM-NB, naive Bayes, and naive Bayes with perfect forgetting. We set DWM-NB to halve the expert weights (i.e., $\beta = 0.5$) and to update these weights and to create and remove experts every fifty time steps (i.e., $p = 50$). We set the algorithm to remove experts with weights less than 0.01 (i.e., $\theta = 0.01$).

In Figure 6, we see the predictive accuracies for DWM-NB, naive Bayes, and naive Bayes with perfect forgetting on the SEA Concepts with 10% class noise. As with the STAGGER Concepts, naive Bayes performed the worst, since it had no method of removing outdated concept descriptions. Naive Bayes with perfect forgetting performed the best and represents the best possible performance for this implementation on this problem. DWM-NB achieved accuracies

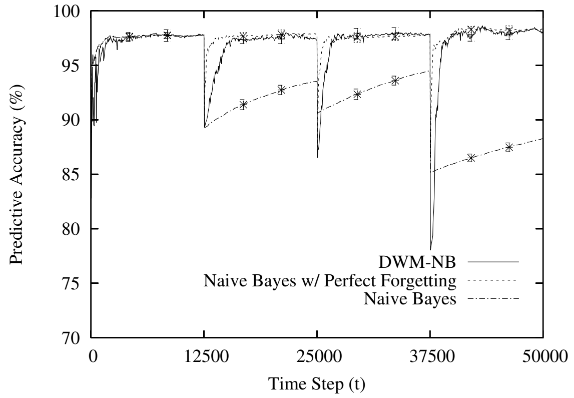

Figure 6. Predictive accuracy for DWM-NB on the SEA Concepts with 10% class noise.

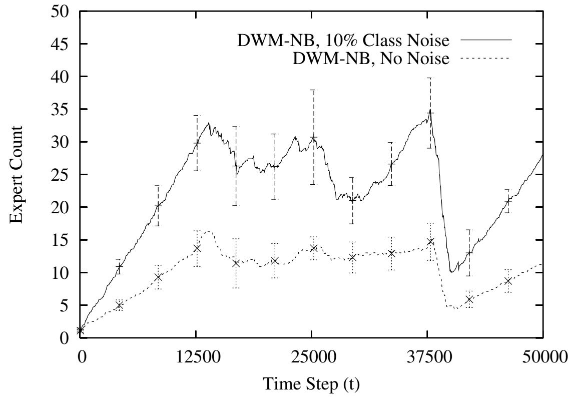

Figure 7. Number of experts maintained for DWM-NB on the SEA Concepts.

nearly equal to those achieved by naive Bayes with perfect forgetting.

Finally, Figure 7 shows the number of experts that DWM-NB maintained during the runs with and without class noise. Recall that DWM creates an expert when it misclassifies an example. In the noisy condition, since 10% of the examples had been relabeled, DWM-NB made more mistakes and therefore created more experts than it did in the condition without noise. In the next section, we analyze these results and place them in context with related work.

5. Analysis and Discussion

In Section 4.1, we presented results for DWM-NB and DWM-ITI on the STAGGER Concepts. In this section, we focus discussion on DWM-ITI, since it performed better than did DWM-NB on this problem. Researchers have built several systems for coping with concept drift and have evaluated many of them on the STAGGER Concepts. For instance, on the first target concept, DWM-ITI did not perform as well as did FLORA2 [26]. However, on the second and third target concepts, it performed notably better than did FLORA2, not only in terms of asymptote, but also in terms of slope.

DWM-ITI and AQ-PM [18] performed identically on the first target concept, but DWM-ITI significantly outperformed AQ-PM on the second and third concepts, again in terms of asymptote and slope. AQ11 [20], although not designed to cope with concept drift, outperformed DWM-ITI in terms of asymptote on the first concept and in terms of slope on the third, but on the second concept, performed significantly worse than did DWM-ITI [19]. Finally, comparing to AQ11-PM [19] and AQ11-PM-WAH [17], DWM-ITI did not perform as well on the first target concept, performed comparably on the second, and converged more quickly on the third.

Overall, we concluded that DWM-ITI outperformed these other learners in terms of accuracy, both in slope and asymptote. In reaching this conclusion, we gave little weight to performance on the first concept, since most learners can acquire it easily and doing so requires no mechanisms for coping with drift. On the second and third concepts, with the exception of AQ11, DWM-ITI performed as well or better than did the other learners. And while AQ11 outperformed DWM-ITI in terms of slope on the third concept, this does not mitigate AQ11's poor performance on the second.

We attribute the performance of DWM-ITI to the training of multiple experts on different sequences of examples. (Weighting experts also contributed, and we will discuss this topic in detail shortly.) Assume a learner incrementally modifies its concept descriptions as new examples arrive. When the target concept changes, if the new one is disjoint, then the best policy to learn new descriptions from scratch, rather than modifying existing ones. This makes intuitive sense, since the learner does not have to first unlearn the old concept, and results from this and other empirical studies support this assertion [18, 19]. Unfortunately, target concepts are not always disjoint, it is difficult to determine precisely when concepts change, and it is challenging to identify which rules (or parts of rules) apply to new target concepts. DWM addresses these problems both by incrementally updating existing descriptions and by learning new concept descriptions from scratch.

Regarding our results for the SEA Concepts [24], which we reported in Section 4.2, DWM-NB outperformed SEA on all four target concepts. On the first concept, performance was similar in terms of slope, but not in terms of asymptote, and on subsequent concepts, DWM-NB converged more quickly to the target concepts and did so with higher accuracy. For example, on concepts 2–4, just prior to the point at which concepts changed, SEA achieved accuracies in the 90–94% range, while DWM-NB's were in the 96–98% range.

We suspect this is most likely due to SEA's unweighted voting procedure and its method of creating and removing new classifiers. Recall that the method trains a new classifier on a fixed number of examples. If the new classifier improves the global performance of the ensemble, then it is added, provided the ensemble does not contain a maximum number of classifiers; otherwise, SEA replaces a poorly performing classifier in the ensemble with the new classifier.

However, if every classifier in the ensemble has been trained on a given target concept, and the concept changes to one that is disjoint, then SEA will have to replace at least half of the classifiers in the ensemble before accuracy on the new target concept will surpass that on the old. For instance, if the ensemble consists of 20 classifiers, and each learns from a fixed set of 500 examples, then it would take at least 5000 additional training examples before the ensemble contained a majority number of classifiers trained on the new concept.

In contrast, DWM under similar circumstances requires only 1500 examples. Assume $p = 500$, the ensemble consists of 20 fully trained classifiers, all with a weight of one, and the new concept is disjoint from the previous one. When an example of this new concept arrives, all 20 classifiers will predict incorrectly, DWM will reduce their weights to 0.5—since the global prediction is also incorrect—and it will create a new classifier with a weight of one. It will then process the next 499 examples.

Assume another example arrives. The original 20 experts will again misclassify the example, and the new expert will predict correctly. Since the weighted prediction of the twenty will be greater than that of the one, the global prediction will be incorrect, the algorithm will reduce the weights of the twenty to 0.25, and it will again create a new expert with a weight of one. DWM will again process 499 examples.

Assume a similar sequence of events occurs: another example arrives, the original twenty misclassify it, and the two new ones predict correctly. The weighted-majority vote of the original twenty will still be greater than that of the new experts (i.e., $20(0.25) > 2(1)$), so DWM will decrease the weight of the original twenty to 0.125, create a new expert, and process the next 499 examples. However, at this point, the three new classifiers trained on the target concept will be able to overrule the predictions of the original twenty, since $3(1) > 20(0.125)$. Crucially, DWM will reach this state after processing only 1500 examples.

Granted, this analysis of SEA and DWM does not take into account the convergence of the base learners, and as such, it is a best-case analysis. The actual number of examples required for both to converge to a new target concept may be greater, but the relative proportion of examples will be similar. This analysis also holds if we assume that DWM replaces experts, rather than creating new ones. Generally, ensemble methods with weighting mechanisms, like those present in DWM, will converge more quickly to target concepts (i.e., require fewer examples) than will methods that replace unweighted learners in the ensemble.

DWM certainly has the potential for creating a large number of experts. We used a simple heuristic that added a new expert whenever the global prediction was incorrect, which intuitively, should be problematic for noisy domains. However, even though on the SEA Concepts DWM-NB maintained as many as 40 experts at, say, time step 37,500, it maintained only 22 experts on average over the ten runs, which is similar to the 20–25 that SEA reportedly stored [24]. If the number of experts were to reach impractical levels, then DWM could simply stop creating experts after obtaining acceptable accuracy; training would continue. Plus, we could also easily distribute the training of experts to processors on a network or in course-grained parallel machine.

One could argue that better performance of DWM-NB is due to differences between the base learners. SEA was an ensemble of C4.5 classifiers [22], while DWM-NB, of course, used naive Bayes as the base algorithm. We refuted this hypothesis by running both base learners on each of the four target concepts. Both achieved comparable accuracies on each concept. For example, on the first target concept, C4.5 achieved 99% accuracy and naive Bayes achieved 98%. Since these learners performed similarly, we concluded that our positive results on this problem were due not to the superiority of the base learner, but to the mechanisms that create, weight, and remove experts.

6. Concluding Remarks

Tracking concept drift is important for many applications. In this paper, we presented a new ensemble method based on the Weighted Majority algorithm [15]. Our method, Dynamic Weighted Majority, creates and removes base algorithms in response to changes in performance, which makes it well suited for problems involving concept drift. We described two implementations of DWM, one with naive Bayes as the base algorithm, the other with ITI [25]. Using the STAGGER Concepts, we evaluated both methods and Blum's implementation of Weighted Majority [3]. To determine performance on a larger problem, we evaluated DWM-NB on the SEA Concepts. Results on these problems, when compared to other methods, suggest that DWM maintained a comparable number of experts, but achieved higher predictive accuracies and converged to those accuracies more quickly. Indeed, to the best of our knowledge, these are the best overall results reported for these problems.

In future work, we plan to investigate more sophisticated heuristics for creating new experts: Rather than creating one when the global prediction is wrong, perhaps DWM should take into account the expert's age or its history of predictions. We would also like to investigate another decision-tree learner as a base algorithm, one that does not maintain encountered examples and that does not periodically restructure its tree; VFDT [6] is a likely candidate. Although removing experts of low weight yielded positive results for the problems we considered in this study, it would be beneficial to investigate mechanisms for explicitly handling noise, or for determining when examples are likely to be from a different target concept, such as those based on the Hoeffding bounds [10] present in CVFDT [11]. We anticipate that these investigations will lead to general, robust, and scalable ensemble methods for tracking concept drift.

Acknowledgments. The authors thank William Headden and the anonymous reviewers for helpful comments on earlier drafts of the manuscript. We also thank Avrim Blum and Paul Utgoff for releasing their respective systems to the community. This research was conducted in the Department of Computer Science at Georgetown University. The work was supported in part by the National Institute of Standards and Technology under grant 60NANB2D0013 and by the Georgetown Undergraduate Research Opportunities Program.

References

[1] B. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[2] C. Blake and C. Merz. UCI Repository of machine learning databases. Web site, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Department of Information and Computer Sciences, University of California, Irvine, 1998.

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

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[5] T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–158, 2000.

[6] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining*, pages 71–80, ACM Press, New York, NY, 2000.

[7] W. Fan, S. Stolfo, and J. Zhang. The application of AdaBoost for distributed, scalable and on-line learning. In *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining*, pages 362–366, ACM Press, New York, NY, 1999.

[8] A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53, 2003.

[9] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, Morgan Kaufmann, San Francisco, CA, 1996.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[11] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining*, pages 97–106, ACM Press, New York, NY, 2001.

[12] G. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345, Morgan Kaufmann, San Francisco, CA, 1995.

[13] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 202–207, AAAI Press, Menlo Park, CA, 1996.

[14] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[15] N. Littlestone and M. Warmuth. The Weighted Majority algorithm. *Information and Computation*, 108:212–261, 1994.

[16] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the 4th National Conference on Artificial Intelligence*, pages 546–551, AAAI Press, Menlo Park, CA, 1997.

[17] M. Maloof. Incremental rule learning with partial instance memory for changing concepts. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2764–2769, IEEE Press, Los Alamitos, CA, 2003.

[18] M. Maloof and R. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.

[19] M. Maloof and R. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, to appear.

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

[21] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

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

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

[24] W. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining*, pages 377–382, ACM Press, New York, NY, 2001.

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

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