Paired Learners for Concept Drift

Stephen H. Bach and Marcus A. Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057-1232 USA
{bach, maloof}@cs.georgetown.edu

Abstract

To cope with concept drift, we paired a stable online learner with a reactive one. A stable learner predicts based on all of its experience, whereas a reactive learner predicts based on its experience over a short, recent window of time. The method of paired learning *uses differences in accuracy between the two learners over this window to determine when to replace the current stable learner, since the stable learner performs worse than does the reactive learner when the target concept changes. While the method uses the reactive learner as an indicator of drift, it uses the stable learner to predict, since the stable learner performs better than does the reactive learner when acquiring a target concept. Experimental results support these assertions. We evaluated the method by making direct comparisons to dynamic weighted majority, accuracy weighted ensemble, and streaming ensemble algorithm (*SEA*) using two synthetic problems, the Stagger concepts and the* SEA *concepts, and three real-world data sets: meeting scheduling, electricity prediction, and malware detection. Results suggest that, on these problems, paired learners outperformed or performed comparably to methods more costly in time and space.*

1. Introduction

Researchers designing algorithms to cope with concept drift have long acknowledged the importance of balancing reactivity and stability. *Concept drift* refers to an online learning task in which the target concept changes over time. Learners for such tasks must be reactive so they adapt quickly to a new target concept, but they must also be stable when acquiring a target concept so as not to be affected by problems typical of online learning, such as ordering effects. A learner that is too reactive may have difficulty acquiring *any* target concept, whereas one that is too stable may be overly burdened by knowledge of a previous concept to learn a new one.

Researchers have approached the problem of concept drift in several ways. The systems they have developed have adjusted and decayed weights (e.g., [14]), maintained and then modified partially learned models (e.g., [6, 17]), maintained previously encountered examples (e.g., [17]), and maintained and consulted multiple models (e.g., [2, 7, 9, 15, 16]). Our work falls into this final category.

Rather than using an ensemble of unweighted or weighted batch learners (e.g., [15, 16]) or one of weighted online learners (e.g., [2, 7, 9]), we paired a stable online learner with a reactive one. The stable learner predicts based on all of its experience. The reactive learner predicts based on its experience over a short, recent window of time. The novel idea is to use the interplay between these two learners and their differences in accuracy to cope with concept drift. The stable learner outperforms the reactive learner when acquiring a target concept, but the reactive learner outperforms the stable learner in the period after the target concept changes. Indeed, when the reactive learner outperforms the stable learner over a short window of time, then the method of *paired learning* replaces the stable learner's knowledge with that of the reactive learner. The pair then continue to learn.

To evaluate paired learners, we conducted an empirical evaluation using five problems that have appeared previously in the literature: the Stagger concepts [14, 17], the SEA concepts [15], the CAP data set [2, 12], electricity prediction [5], and malware detection [8]. We compared the paired learners to four methods: a single base learner, streaming ensemble algorithm [15], dynamic weighted majority [9], and accuracy weighted ensemble [16]. Results suggest that, on the five problems, paired learners outperformed or performed comparably to learners more costly in time and space. In some cases, these methods required between 10 and 50 base learners to obtain high accuracy on the problems considered, but our method used two. Ironically, for one problem, we obtained the best performance for two methods when their ensembles had two members.

We find the notion of paired learning appealing because

of how directly the method addresses desiderata for learners for concept drift: reactivity and stability. While present in all algorithms—in weighting procedures, in maintaining alternative hypotheses, in storing previous examples from the stream—paired learning achieves these desiderata explicitly by using a reactive learner and a stable learner. We anticipate that the method's simplicity and the clarity it provides will give researchers additional insight into the problem of learning concepts that change over time. Indeed, since paired learning stands in marked contrast to recently proposed ensemble methods that require upwards of 50 base learners to achieve high accuracy on commonly used data sets, the outcome of this study has implications not only for the design of algorithms for concept drift, but also for the problems used to evaluate such algorithms.

2. Related work

Researchers have introduced a number of ensemble methods for concept drift. We review those algorithms included in our experimental study, which we selected because, like paired learning, the base learners train on different sequences of examples from the stream. The streaming ensemble algorithm [15], or SEA, maintains a fixed-capacity, unweighted collection of m batch learners. SEA builds a classifier using a batch of p examples. In addition to the classifiers in the ensemble, it maintains two classifiers in reserve, C_i, constructed from the current batch, and C_{i-1}, constructed from the previous batch. If space exists in the ensemble, then SEA adds C_{i-1}. Otherwise, SEA replaces a poorer performing classifier in the ensemble with C_{i-1}, as measured on the current batch, provided that such a classifier exists. SEA predicts the majority prediction of the members of the ensemble. Using C4.5 [13] as the base learner, the authors evaluated SEA on a problem of their design, which we refer to as the "SEA concepts."

Dynamic weighted majority [9], or DWM, maintains a dynamically sized, weighted collection of online learners. It predicts based on a weighted-majority vote of the learners' predictions and decreases the weights of those learners that predict incorrectly. If the algorithm's global prediction is incorrect, then it adds a new online learner to the ensemble. Also, it removes a learner if its weight falls below a threshold. Consequently, the size of the ensemble m can vary. Its parameter p specifies the period over which it trains the members of the ensemble, but does not update weights or add or remove learners. The authors evaluated DWM with naive Bayes as the base learner on the Stagger concepts [14] and on the SEA concepts [15]. (See Section 4).

Accuracy-weighted ensemble [16], or AWE, maintains a fixed-capacity, weighted collection of m batch learners. AWE builds a classifier from a batch of p examples and computes the error rate of the classifier on the examples using cross-validation. It then derives a weight for the classifier using either costs or mean-squared error; our implementation uses the latter. AWE then evaluates each member of the ensemble on the new batch and adjusts its weight. It forms a new ensemble by storing the m highest-weighted classifiers. AWE predicts the weighted-majority prediction of the members of the ensemble. Using credit-card fraud data and a shifting hyperplane of their own design, the authors conducted an extensive evaluation that included using AWE with naive Bayes as the base learner.

Other learners for concept drift include Stagger [14], the FLORA systems [17], winnow [2, 10], weighted majority [2, 11], the concept-drifting very fast decision tree learner [6], additive experts [7],[1] ultra-fast forest of trees [4], and online rank [1].

3. An algorithm for paired learning

A paired learner (Algorithm 1) consists of the stable learner S that predicts based on all of its experience and the reactive learner R_w that predicts based on its most recent experience over a window of length w. It uses S to predict and uses R_w as a lagging indicator of drift. Input to the algorithm (line 1) is a collection of T training examples, the window's length w, and the threshold θ for creating a new stable learner. To classify an instance, the algorithm uses S's prediction as its prediction (lines 9 & 10). Learning entails passing a new example to the learning elements of S and R_w (lines 22 & 23). To cope with concept drift, at time t, if S incorrectly classifies an example and R_w correctly classifies it (line 12), then the paired learner sets bit t in the circular list C of w bits (line 13); otherwise, it unsets the bit (line 15). If the proportion of bits set in C surpasses the threshold θ (line 17), then the paired learner replaces S with a new stable learner (line 18) and sets its initial concept description to that of R_w (line 19). It also clears the bits of C (line 20).

A learner for concept drift must incorporate new examples into its model when the target concept is stable and replace an outdated model or portions of the model when the target concept changes. We designed the algorithm for paired learning to detect change by comparing the performance of a stable learner to that of a reactive one. Since the only difference between the two is that the stable learner has learned from all examples since the last replacement and the reactive learner has learned from the w most recent examples, when the reactive learner outperforms the stable one, it suggests that the examples older than $t - w$ are *harming* the performance of the ensemble.

The method attempts to determine a recent point in the data stream at which the target concept changed so the

[1] This method is similar to DWM; in practice, we have found that DWM can perform better, so we used it in this study.

learner can focus on acquiring a new model with the most recently received examples. To do so, however, we discovered through pilot studies that it is not sufficient to consider only the accuracy of S and R_w on the last w examples. To illustrate, consider the case in which S correctly classifies the first $\frac{w}{2}$ examples, misclassifies the next $\frac{w}{2}$ examples, and R_w does the opposite (i.e., classifies the first $\frac{w}{2}$ incorrectly, and classifies the next $\frac{w}{2}$ correctly). Both S and R_w have an accuracy of 50% over the window, but their comparative performance indicates that the accuracy of S is decreasing and the accuracy of R_w is increasing. This suggests that the target concept may have changed, but accuracy gives no information to this effect.

To identify such situations, paired learning measures how frequently S misclassifies an example that R_w classifies correctly over the last w examples. If this event occurs frequently enough over w, then the method replaces S with R_w and learning continues. This replacement mechanism is advantageous because it preserves in S what R_w has learned from the last w examples, and as a consequence, the method can use S immediately to classify observations. Crucially, in contrast to methods that accumulate batches of examples, paired learning can apply this mechanism at any point in the stream.

Note that one can use any online algorithm as a base learner for paired learning. For this study, we used naive Bayes, which we discuss in the next section.

Algorithm 1 Paired Learner

Input: $\{\vec{x}_t, y_t\}_{t=1}^{T}$, w, θ
$\{\vec{x}_t, y_t\}_{t=1}^{T}$: training data
w: window size for the reactive learner
θ: threshold for creating a new stable learner
Let S be a stable learner
Let R_w be a w-reactive learner
Let C be a circular list of w bits, each initially 0
for $t \leftarrow 1$ **to** T **do**
  $\hat{y}_S \leftarrow S.\mathsf{Classify}(\vec{x}_t)$
  output $\hat{y}_S$
  $\hat{y}_R \leftarrow R_w.\mathsf{Classify}(\vec{x}_t)$
  if $\hat{y}_S \neq y_t \wedge \hat{y}_R = y_t$ **then**
    $C.\mathsf{set}(t)$
  else
    $C.\mathsf{unset}(t)$
  end if
  if $\theta < C.\mathsf{proportionOfSetBits}()$ **then**
    $S \leftarrow$ **new** $\mathsf{StableLearner}()$
    $S \leftarrow R_w.\mathsf{getConceptDescription}()$
    $C.\mathsf{unsetAll}()$
  end if
  $S.\mathsf{Train}(\vec{x}_t, y_t)$
  $R_w.\mathsf{Train}(\vec{x}_t, y_t)$
end for

3.1. Paired learner with naive Bayes

We implemented a paired learner using an online version of naive Bayes (NB) as the base learner. As its concept description, NB stores distributions for each class, C_i, and for each attribute given the class. For symbolic attributes, it stores frequency counts of each value. For numeric attributes, it stores the sum of the values and the sum of the squared values, under an assumption of normality. The learning element estimates these distributions from training data. Given the instance $\vec{x}$, the performance element uses the distributions to compute prior and conditional probabilities for each class, $P(C_i)$ and $P(x_j|C_i)$, assumes attributes are conditionally independent, and uses Bayes' rule to predict the most probable class C:

$$C = \operatorname*{argmax}_{C_i} P(C_i) \prod_{j=1}^{n} P(x_j|C_i) \ .$$

We refer to this learner as PL-NB, and we implemented two versions using WEKA [18]. Both implementations use as the stable learner the online version of NB, but their reactive learners are different. The first rebuilds the learner's model with the arrival of each new training example from it and last $w-1$ examples. An advantage of this scheme is that it is general and works for all learners (batch and online). A disadvantage is that running time increases with w, but this is also true for SEA [15] and AWE [16] and the parameter p.

The second version uses as the reactive learner what we call a *retractable learner*, for it can unlearn an example. The retractable learner also maintains w examples, but when a new example arrives, the learner retracts the last example in the window from the model, removes the example from the window, and then adds the new example to the model and to the window.

For naive Bayes, these operations are easy and efficient to implement and involves subtracting and adding counts and values from and to the appropriate distributions. An advantage of the retractable learner is that it is significantly faster than the first version. (We discuss timing and complexity issues in Sections 5.1 and 5.2.) The disadvantage is that it is not general, for not all online methods will have a retractable version. Both versions perform identically in terms of accuracy and require the same amount of memory.

4. Experimental study

To evaluate paired learning for concept drift, we conducted a number of experiments involving three other ensemble methods for concept drift and five problems and data

sets that have appeared previously in the literature. For the base learner, as mentioned, we implemented an online version of NB. For ensemble methods, we implemented SEA [15], DWM [9], and AWE [16]. To identify an ensemble method and base learner, we write the method's name followed by the base learner's name. For example, SEA-NB refers to SEA with NB as the base learner.

For the purpose of experimentation, the choice of naive Bayes as the base learner is an important one. SEA [15] and AWE [16] use batch methods as their base learners, whereas paired learners and DWM [9] use online methods. When evaluating and comparing ensemble methods, it is important to control for confounding experimental factors due to the base learners. The advantage of using naive Bayes is that its batch and online versions produce the same model from a given set of training examples irrespective of their order, since the addition operations that update its model are commutative.

In the following sections, we organize discussion around problems and data sets, which we selected because they have been used in previous evaluations of learners for concept drift. The malware data set is an exception, but we included it as another example of a real-world data set. The real-world data sets have no ground truth, so we cannot make strong claims about the presence or type of drift, although correlational evidence suggests that the drops in performance for the CAP data set correspond to semester boundaries [12]. Nonetheless, we concluded that the benefit of evaluating the methods on these real-world data sets justified their inclusion.

We present only the best results for each method, and determined the best performing parameters using a grid search. We searched over two parameters by defining ranges and increments for each method and its parameters. We ran the methods on all combinations of these parameters, and measured performance using accuracy or area under the performance curve, as described in the following sections. To better understand the parameter space, we plotted three-dimensional graphs and used this information to expand the grid over which we searched if it seemed that performance might improve. Naturally, we make no claims that these parameters are optimal, but they are consistent with those published previously. In one instance, we found better parameter settings than did the original authors.

4.1. Stagger concepts

The Stagger concepts [14, 17] consist of three attributes, each taking three values, and three target concepts presented over 120 time steps. The attributes and their values are *shape* $\in \{$*triangle, circle, rectangle*$\}$, *color* $\in \{$*red, green, blue*$\}$, and *size* $\in \{$*small, medium, large*$\}$. For the first 40 time steps, the target concept is *color* $=$ *red* $\wedge$ *size* $=$ *small*.

Table 1. Results for the Stagger concepts. Measures are average normalized area under the curve (AUC) after the first drift point and 95% confidence intervals.

Learner and Parameters	AUC
NB, on each concept	0.914±0.007
DWM-NB, $p = 1$, $m = 5$	0.868±0.007
PL-NB, $\theta = .1$, $w = 6$	0.865±0.010
AWE-NB, $p = 10$, $m = 5$	0.808±0.010
SEA-NB, $p = 8$, $m = 2$	0.732±0.011
NB, on all examples	0.516±0.011

For the next 40 time steps, the target concept is *color* $=$ *green* $\vee$ *shape* $=$ *circle*. And for the last 40 time steps, the target concept is *size* $=$ *medium* $\vee$ *size* $=$ *large*.

While this may seem like a simple problem, we must note that the first target concept is conjunctive, the second is disjunctive, and the third is internally disjunctive (i.e., one attribute takes multiple values). Moreover, the first and second target concepts share only one positive example, and so it is almost a concept reversal. Crucially, it is not always the size of the feature space or the number of examples that makes a learning problem difficult. A number of researchers have used this problem to evaluate methods of concept drift (e.g., [7, 9, 14, 17]).

At each time step, one presents a single, random example to the learner and then tests it on a set of 100 random examples. We evaluated each algorithm using this protocol, repeated 50 times. We measured accuracy on the test set and computed average accuracy and 95% confidence intervals at each time step.

We also computed the average normalized area under the performance curves (AUC) *after* the first drift point (with 95% confidence intervals). We computed AUC using the trapezoid rule on adjacent pairs of accuracies and normalized by dividing by the total area of the region. The areas under the entirety of the curves were similar, but we chose to compute AUC after the first drift point because most learners perform well on the first concept, and we are most interested in performance after drift occurs. Although AUC is a convenient measure, it may not represent important aspects of performance, such as slope and asymptote, and so we center discussion on the plots of the performance curves.

Figure 1 shows the effect of paired learning on naive Bayes' performance. We compared PL-NB to NB trained on all examples and on examples of each individual concept. These latter two conditions represent the worst- and best-case scenarios, respectively. The areas under these curves appear in Table 1. PL-NB (with an AUC of .865) performed almost as well as NB trained on each individual concept

(with an AUC of .914) and performed much better than did NB trained on all examples (with an AUC of .516).

That the paired learner outperformed the base learner trained on all examples is not surprising, but one must keep in mind that, compared to the base learner, which is a stable learner, the paired learner adds only a reactive learner and basic control policies for replacing the stable learner's knowledge with that of the reactive learner. We found these results encouraging and supportive of our approach.

Figure 2 shows the best performances for PL-NB, DWM-NB, SEA-NB, and AWE-NB. In terms of accuracy, PL-NB performed comparably to DWM-NB, the best performing method overall. AWE-NB and SEA-NB did not perform as well as PL-NB on this problem because they had to wait to accumulate examples into a batch before learning. Observe the "stair steps" in their performance curves. Smaller values for p did not improve performance. Also, notice that SEA-NB performed best when its ensemble had two members.

4.2. SEA concepts

To evaluate SEA, Street and Kim [15] introduced what we call the SEA concepts, which consist of four target concepts presented over 50,000 time steps. The target concept changes every 12,500 time steps, and one generates a single test set for each *concept* consisting of 2,500 examples. There is 10% class noise for the training examples, and we evaluated the learners every 100 time steps. Each example consists of numeric attributes $x_i \in [0, 10]$, for $i = 1, \ldots, 3$. The target concepts are hyperplanes, such that $y = +$ if $x_1 + x_2 \leq \theta$, where $\theta \in \{7, 8, 9, 9.5\}$ for each of the four target concepts, respectively; otherwise, $y = -$. Thus, x_3 is an irrelevant attribute. We conducted 10 trials and averaged accuracy on the test set. We also computed the average normalized area under the curve after the first drift point. We computed 95% confidence intervals for both measures. (Several researchers have used a shifting hyperplane to evaluate learners for concept drift [6, 7, 9, 15, 16].)

Table 2 presents the learners, their parameters, and their AUCs with 95% confidence intervals. In Figure 3, we present results for PL-NB, NB trained on all examples, and NB trained on the examples of each individual concept. PL-NB performed as well as did NB trained on each concept. As shown in Figure 4, all of the methods performed well. There is virtually no difference in the performances of PL-NB, SEA-NB, and NB trained on each concept (see Table 2). Notice, however, that DWM-NB performed well, but required 50 base learners to achieve this performance, whereas PL-NB required only two. Also, SEA-NB with two base learners resulted in notably better performance on this task than that reported by Street and Kim [15]. Indeed, that SEA-NB and AWE-NB performed best with two members in their ensembles is supportive of our approach.

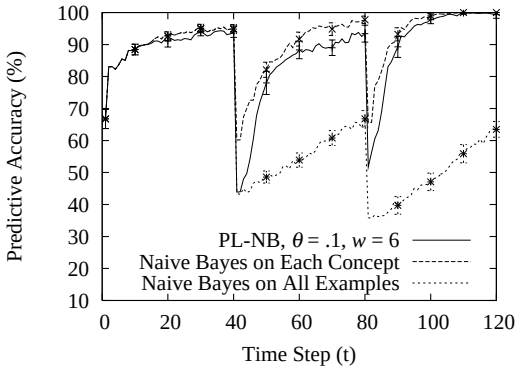

Figure 1. Results for PL-NB on the Stagger concepts. Measures are accuracy with 95% confidence intervals.

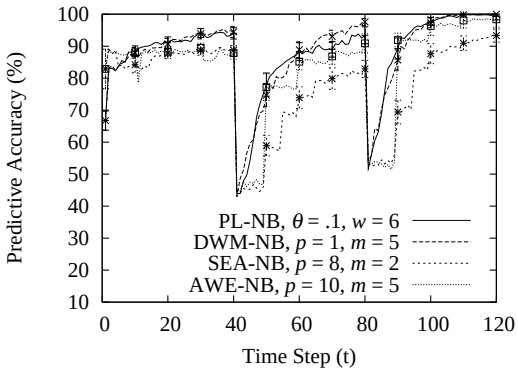

Figure 2. Results for PL-NB, SEA-NB, DWM-NB, and AWE-NB on the Stagger concepts. Measures are accuracy with 95% confidence intervals.

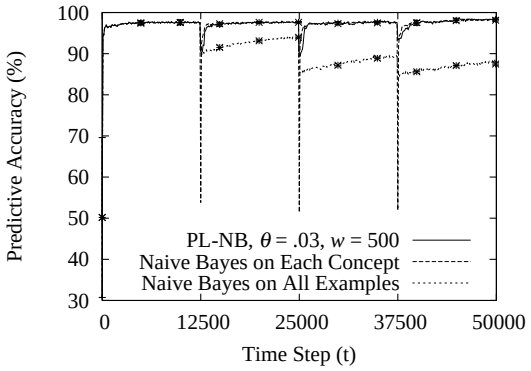

Figure 3. Results for PL-NB on the SEA concepts. Measures are accuracy with 95% confidence intervals.

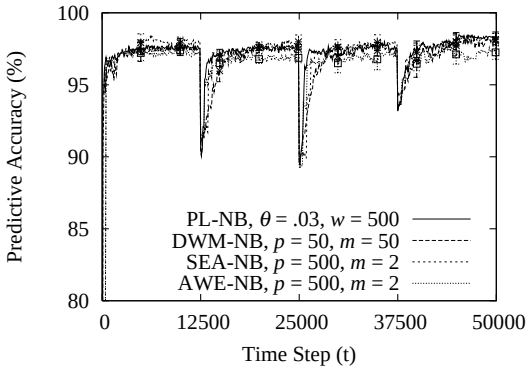

Figure 4. Results for PL-NB, SEA-NB, DWM-NB, and AWE-NB on the SEA concepts. Measures are accuracy with 95% confidence intervals.

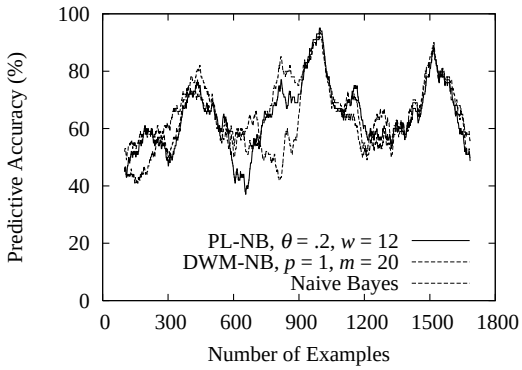

Figure 5. Accuracy on the duration task for PL-NB, NB, and DWM-NB. Measures are averages of the previous 100 predictions.

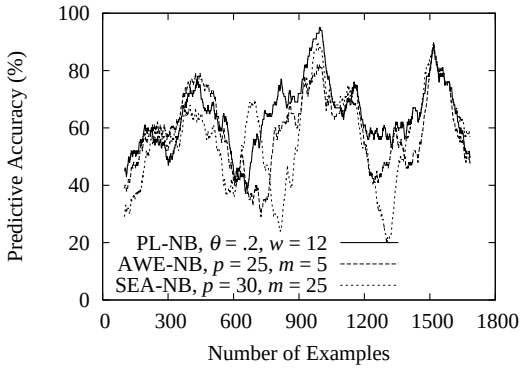

Figure 6. Accuracy on the duration task for PL-NB, SEA-NB, and AWE-NB. Measures are averages of the previous 100 predictions.

Table 2. Results for the SEA concepts. Measures are average normalized area under the curve (AUC) after the first drift point and 95% confidence intervals.

Learner and Parameters	AUC
PL-NB, $\theta = .03$, $w = 500$	0.972±0.003
SEA-NB, $p = 500$, $m = 2$	0.971±0.002
NB, on each concept	0.971±0.001
DWM-NB, $p = 50$, $m = 50$	0.969±0.002
AWE-NB, $p = 500$, $m = 2$	0.966±0.002
NB, on all examples	0.890±0.002

4.3. Calendar scheduling

For a real-world application, we evaluated the methods on the calendar-apprentice (CAP) data set [2, 12], a calendar-scheduling task. Based on a subset of 34 symbolic attributes, the task is to predict a user's preference for a meeting's location, duration, start time, and day of week. There are 12 attributes for location, 11 for duration, 15 for start time, and 16 for day of week. We processed online the 1,685 examples for User 1 by testing on each new example, measuring accuracy, and then using it to train each learner.

We present the results for the CAP data set in Table 3. Overall, PL-NB performed comparably to DWM-NB and outperformed the other methods. DWM-NB performed slightly better than did PL-NB in terms of accuracy, but required 20 base learners to do so. Not only did DWM require more memory than did PL-NB, but also it required considerably more time, since it trained each of the 20 learners in the ensemble on each new instance.

Figure 5 shows the performance curves on the duration task for PL-NB, NB, and DWM-NB, and Figure 6 shows these curves for PL-NB, AWE-NB, and SEA-NB. We suspect that PL-NB, NB, and DWM-NB had an advantage on this task since they immediately learned from each new example, instead of accumulating examples into a batch, as did SEA-NB and AWE-NB. We did try smaller ensembles and shorter periods for these learners, but the settings did not produce higher accuracy on this task.

4.4. Electricity prediction

The electricity-prediction data set consists of 45,312 examples collected at 30-minute intervals between 7 May 1996 and 5 December 1998 [5]. The task is to predict whether the price of electricity will go up or down based on five numeric attributes: the day of the week, the 30-minute period of the day, the demand for electricity in New South

Table 3. Accuracy on the CAP data set using 1,685 examples for User 1.

Prediction Task	NB	PL-NB $\theta = .2, w = 12$	SEA-NB $p = 30, m = 25$	DWM-NB $p = 1, m = 20$	AWE-NB $p = 25, m = 5$
Location	62.93	66.16	58.95	66.90	58.36
Duration	63.25	65.13	59.29	65.90	59.56
Start Time	33.24	38.19	27.07	38.87	27.31
Day of Week	52.29	51.48	40.95	51.70	40.58
Average	52.92	55.24	46.57	55.84	46.17

Table 4. Results for the electricity-prediction task. Measures are accuracy averaged over 45,312 predictions.

Learner and Parameters	Accuracy (%)
DWM-NB, $p = 1, m = 5$	82.81
PL-NB, $\theta = .8, w = 1$	81.26
AWE-NB, $p = 18, m = 5$	65.84
SEA-NB, $p = 72, m = 5$	64.89
NB	62.34

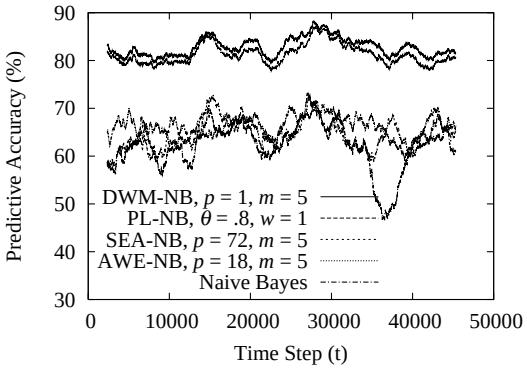

Figure 7. Results for the electricity-prediction task. Measures are accuracy averaged over the previous 2,352 predictions.

Wales, the demand in Victoria, and the amount of electricity to be transferred between the two. Roughly 39% of the examples have unknown values for either demand in Victoria or the transfer amount.

To evaluate the methods, we processed the examples online in their temporal order by testing on a new example, measuring accuracy, and then training on the example. We computed average accuracy, which we present in Table 4, and we produced performance curves by plotting the average accuracy of the previous 2,352 predictions. These curves appear in Figure 7.

As one can see, PL-NB performed almost as well as DWM-NB did on this task. Both achieved overall accuracies above 80%. Both required reactive parameter settings: $w = 1$ for PL-NB and $p = 1$ for DWM-NB. (For PL-NB, since $w = 1$, θ was irrelevant.) SEA-NB and AWE-NB, again, may have been disadvantaged on this task because they accumulated examples into a batch for learning. Indeed, if the environment changes quickly enough, it may change before such a learner can produce a new batch for learning. One could reduce the size of the batch, but such a reduction may result in poor performing classifiers. Although DWM-NB performed slightly better than did PL-NB, PL-NB maintained only two learners, whereas DWM-NB maintained five. Finally, notice the large dip in NB's performance around time step 38,000. The methods designed for concept drift were either unaffected (e.g., SEA-NB and AWE-NB) or affected little (e.g., PL-NB and DWM-NB) by the phenomenon that produced this drop in NB's accuracy.

4.5. Malware detection

The malware-detection task involves determining whether n-grams extracted from executables are from a benign executable or a malicious executable [9]. There are 3,622 examples, and each consists of 500 Boolean attributes indicating the presence or absence of a 4-gram of bytes. We processed the examples online by shuffling the order of the examples, testing on each example, measuring accuracy, and training on the example. We repeated this process 200 times for each of the five methods.

In Table 5, we present average accuracy, and Figure 8 shows performance curves for the five methods. As shown, all of the methods performed comparably, but PL-NB was more efficient in terms of space. Notice that SEA-NB, DWM-NB, and AWE-NB maintained 10 base learners and that SEA-NB and AWE-NB each accumulated a batch of 50 examples for learning. In contrast, PL-NB maintained 2 base learners and only 9 examples.

We found it interesting that PL-NB and NB achieved the same accuracy on this problem. Subsequent analysis confirmed that, over the 200 trials, PL-NB never replaced its stable classifier, and so the two methods produced identical results. While this suggests an absence of drift and sampling effects, we consider it a positive outcome that, on such a task, PL-NB performed no worse than did its base learner.

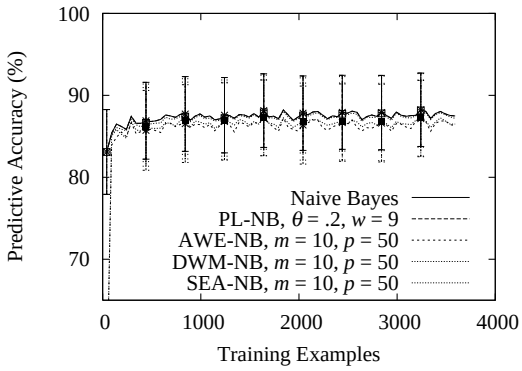

Figure 8. Results for the malware-detection task. Measures are accuracy averaged over the previous 40 predictions.

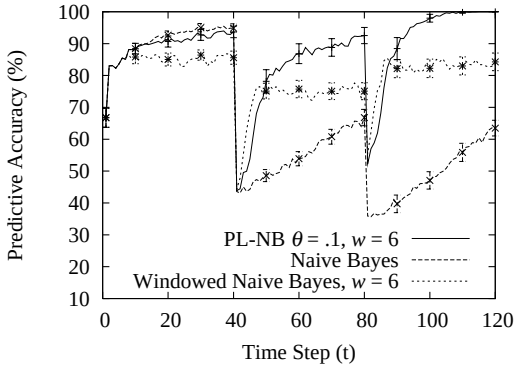

Figure 9. Results for PL-NB, stable NB, and windowed NB on the Stagger concepts.

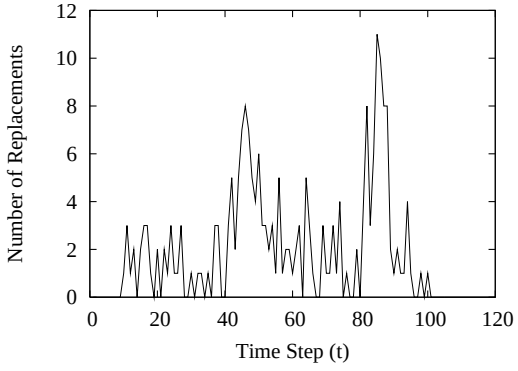

Figure 10. Number of replacements for PL-NB on the Stagger concepts over 50 runs.

Table 5. Results for the malware-detection task. Measures are accuracy with 95% confidence intervals

Learner and Parameters	Accuracy (%)
NB	87.4±0.1
PL-NB, $\theta = .2, w = 9$	87.4±0.1
DWM-NB, $p = 50, m = 10$	87.2±0.1
SEA-NB, $p = 50, m = 10$	86.0±0.2
AWE-NB, $p = 50, m = 10$	85.7±0.2

5. Analysis and discussion

The paired learner is one of the most direct and efficient algorithms proposed thus far for coping with concept drift. It consists of two online learners, a stable learner and a reactive learner. The paired learner uses the stable learner's prediction as its global prediction. The algorithm creates a new stable learner when, over a short window of time, the reactive learner is correct and the stable learner in incorrect sufficiently often. The method performed comparably to or better than other ensemble methods designed for concept drift on five different problems, and generally did so in less time and space.

Paired learners performed well on the problems considered because its combination of stable and reactive learners is an accurate lagging indicator of concept drift. To support this assertion, we consider the algorithm's performance on the Stagger concepts, which allows for careful analysis. Figure 9 shows the accuracy over 50 runs of PL-NB, windowed NB, and NB trained on all examples. Clearly, the standard version of NB trained on all examples cannot cope with changes in the target concept and performs poorly after the first drift point. In contrast, the windowed version of NB is too reactive and learns a model with only limited accuracy. However, both versions have advantages for different parts of the problem. The standard version is better at learning stable (i.e., individual) concepts (see Figure 1), and the reactive version is better at learning immediately after a drift point.

We contend that a reactive learner outperforming a stable learner is rare, *unless* drift has occurred. Our experimental results support this assertion, but to illustrate further, Figure 10 shows the number of times that PL-NB replaced S at each time step on the Stagger concepts. Note that if a replacement occurs at time step t, then the new stable learner S will learn from the examples received from time step $t - w + 1$ onward. The optimal replacement times for $w = 6$ on the Stagger concepts are $t = 46$ and $t = 86$ (with no other replacements). As one can see, PL-NB tended to replace S around these optimal times. If the paired learner replaces S near these times, then it should classify subse-

quent examples more accurately. Figures 1, 9, and 10 show that PL-NB more closely approached optimal replacement performance on the third target concept than on the second, which resulted in accuracy closer to that of the base learner trained on each concept.

5.1. Timing results

To characterize the relative running times of the methods in our study, we measured combined training and testing times on the SEA concepts, the most time-consuming task considered. All problem and learning parameters are the same as those reported in Section 4.2. We performed the evaluations on a workstation with a two-core, 2.8 GHz Intel Pentium D CPU with 3.5 GB of RAM using Sun's Java Runtime Environment 6 and WEKA version 3.5.7.

For PL-NB, the average running time for five trials was 59.8 seconds. When PL-NB used a retractable learner for R_w, rather than retraining, the average running time was 12.3 seconds. The average running time over five trials for AWE-NB was 24.5 seconds, for SEA-NB was 29.5 seconds, and for DWM-NB was 204.9 seconds. These timing results are consistent with those we observed for other problems.

5.2. Complexity analysis

Let $f(n)$ be the running time required to train a base learner on n examples, and let $g(n)$ be the running time required for that base learner to classify n observations. The time to train a paired learner on n examples is $O(n \cdot f(w) + n \cdot g(1))$, where w is the size of R_w's window. If the base learner retracts examples, then the time required is $O(n \cdot f(1) + n \cdot g(1))$. The time to classify n instances for both versions is $O(n \cdot g(1))$.

For SEA, the training time is $O(\frac{n}{p} \cdot f(p) + \frac{mn}{p} \cdot g(p))$, and the classification time is $O(mn \cdot g(1))$, where m is the number of experts in the ensemble, and p is the number of examples in each batch. For DWM, the training time $O(mn \cdot f(1) + mn \cdot g(1))$, whereas the classification time is $O(mn \cdot g(1))$. Finally, for AWE, training requires $O(\frac{n}{p} \cdot f_{\text{cv}}(p) + \frac{mn}{p} \cdot g(p))$, where $f_{\text{cv}}(n)$ is the time required to train the base learner on n examples using cross-validation. Its classification time is $O(mn \cdot g(1))$.

In terms of space, if $f(n)$ is the space required for a base learner to store a model built from n examples, then the space required for paired learning is $O(f(n) + w)$. DWM requires $O(m \cdot f(n))$, and SEA and AWE require $O(m \cdot f(p) + p)$. We should note that the bound for DWM is overly pessimistic for base learners with models that grow with n. DWM dynamically adds and removes learners, and so it is unlikely that any base learner will learn from all n examples. However, this bound is accurate for base learners with constant memory, such as naive Bayes.

5.3. Results with other base learners

As we have mentioned, one can use any online learner for paired learning. We have evaluated paired learning with other base learners, such as Hoeffding trees (HT), an online method for building decision trees [3]. On the Stagger concepts, DWM-HT ($p = 1, m = 10$) achieved an AUC of .851, but PL-HT had an AUC of .835. On the CAP data set, DWM-HT ($p = 1, m = 10$) achieved 50.68%, whereas PL-HT ($\theta = .4, w = 12$) obtained 55.83%. On the malware data set, PL-HT ($\theta = .2, w = 9$) obtained 90.3%, whereas SEA-J48 ($p = 50, m = 10$) achieved 92.9%, HT alone achieved 92.9%, and AWE-J48 ($p = 50, m = 10$) achieved 92.7%. (J48 is the implementation of C4.5 [13] in WEKA [18].)

As before, the paired learner achieved higher or comparable accuracy, while requiring less time and space than the other methods. We plan to expand our study to include additional ensemble methods, base learners, and data sets in an effort to better characterize and understand trade-offs in performance between paired learning and other methods for concept drift.

5.4. Robustness to noise and varying drift points

We conducted two additional experiments that were interesting, but require further study, and we plan to report more detailed conclusions in future publications. Firstly, we were concerned that the paired learner would be more sensitive to noise than would SEA, DWM, and AWE, even though we saw no such sensitivity to the 10% noise present in the SEA concepts and in other problems. To investigate, we varied the level of class noise from 0% to 70% in increments of 10% for both the Stagger concepts and the SEA concepts, measuring AUC. As expected, the performance of all of the methods decreased with increasing noise, but they degraded at roughly the same rate, suggesting that PL-NB was equally robust to noise.

Secondly, we were interested in the extent to which the relationship between the parameter settings of the algorithms and the drift points of the problems affected accuracy. In evaluations, critical parameters, such as p or w, are often factors of the problem's drift points, which could influence a method's overall accuracy. We investigated by selecting the Stagger concepts [14] and varying over all combinations of drift points in $\{20, \ldots, 60\}$ for the first drift point and in $\{60, \ldots, 100\}$ for the second. (The drift points for the Stagger concepts are 41 and 81.) We ran PL-NB and the other methods, and the accuracy of these methods was generally consistent across all combinations of drift points. The relatively short periods between the drift points may have been insufficient to produce a measurable effect, and we plan to study this issue further with other problems.

We contend that the paired learner is of less concern

in this regard since its window slides. Moreover, components of the paired learner are not "dormant" as some are in SEA, DWM, and AWE. For example, DWM does not update weights or add or remove experts during its period, and SEA and AWE accumulate examples during periods to build new classifiers. We intend to develop further this experiment and report its results elsewhere.

6. Concluding remarks

We introduced the notion of a paired learner, which uses the difference in performance between a reactive learner and a stable learner to cope with concept drift. It has a minimal ensemble in that it maintains and trains only two learners, in contrast to other ensemble methods that use an unweighted or weighted collection of batch learners or a weighted collection of online learners.

We anticipate that the method's directness and the clarity it provides will give researchers further insight into the problem of learning concepts that change over time. We hope the outcome of this study has implications not only for the design of algorithms for concept drift, but also for the problems used to evaluate such algorithms. Results on five previously published problems suggest that paired learners perform comparably to or better than DWM, SEA, and AWE, while requiring less time and space.

Questions remain about ensemble methods for concept drift. We hope to better understand the problems for which large weighted and unweighted ensembles are warranted, although it is not clear if problems or data sets presently exist to support such an investigation. We also plan to study further the properties of reactive learners and their use as indicators of concept drift. We will expand our empirical investigation to include single-model methods for concept drift, such as winnow and Stagger, and other instance generators, such as that used to evaluate AWE. We have already begun to characterize how paired learners and other methods respond to varying drift points and varying amounts of class noise; preliminary results are encouraging.

Acknowledgments. The authors thank Clay Shields and the anonymous reviewers for helpful comments on earlier drafts of this paper. This research was supported in part by GUROP, the Georgetown Undergraduate Research Opportunities Program. The authors are listed in alphabetical order.

References

[1] H. Becker and M. Arias. Real-time ranking with concept drift using expert advice. In *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 86–94. ACM Press, New York, NY, 2007.

[2] A. Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

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

[4] J. Gama, P. Medas, and P. Rodrigues. Learning decision trees from dynamic data streams. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC-2005)*, pages 573–577. ACM Press, New York, NY, 2005.

[5] M. Harries, C. Sammut, and K. Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.

[6] 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*, pages 97–106. ACM Press, New York, NY, 2001.

[7] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456. ACM Press, New York, NY, 2005.

[8] J. Z. Kolter and M. A. Maloof. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 7:2721–2744, 2006.

[9] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007.

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

[11] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[12] T. M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80–91, 1994.

[13] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

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

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

[16] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM Press, New York, NY, 2003.

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

[18] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005.