Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Incremental Learning in Nonstationary Environments with Controlled Forgetting

Ryan Elwell and Robi Polikar*

Abstract — **We have recently introduced an incremental learning algorithm, called Learn^{++}.NSE, designed for *N*on-*S*tationary *E*nvironments (concept drift), where the underlying data distribution changes over time. With each dataset drawn from a new environment, Learn^{++}.NSE generates a new classifier to form an ensemble of classifiers. The ensemble members are combined through a dynamically weighted majority voting, where voting weights are determined based on classifiers' age-adjusted accuracy on current and past environments. Unlike other ensemble-based concept drift algorithms, Learn^{++}.NSE does not discard prior classifiers, allowing potentially cyclical environments to be learned more effectively. While Learn^{++}.NSE has been shown to work well on a variety of concept drift problems, a potential shortcoming of this approach is the cumulative nature of the ensemble size. In this contribution, we expand our analysis of the algorithm to include various ensemble pruning methods to introduce controlled forgetting. Error or age-based pruning methods have been integrated into the algorithm to prevent potential outvoting from irrelevant classifiers or simply to save memory over an extended period of time. Here, we analyze the tradeoff between these precautions and the desire to handle recurring contexts (cyclical data). Comparisons are made using several scenarios that introduce various types of drift.**

***Index Terms*—concept drift, learning in nonstationary environments, multiple classifier systems, incremental learning**

I. Introduction

Incremental learning from data drawn from a non-stationary environment is an increasingly important topic in computational intelligence due to many potential applications – from climate or financial data analysis to monitoring network traffic – that can benefit from such a capability. In a nonstationary environment, the underlying distribution that generates the data, and therefore the corresponding decision boundaries, changes over time. The classification algorithm must then be able to adapt such that the learned decision boundaries can be updated accordingly. However, the classifier should also retain any previously acquired knowledge that is still relevant, which raises the stability-plasticity dilemma [1]. Learning in such an environment becomes particularly challenging if the previous data are no longer available, requiring an incremental approach, where learning must rely on existing models (classifiers) and the current data only [2].

Early work on learning in nonstationary environments have primarily focused on the definition of the problem, as well as the conditions under which such learning is possible, for example, within the PAC (probably approximately correct) learning framework [3-5]. More recently, non-stationary environments have been modeled in more pragmatic terms, such as real or virtual drift (change in class conditional or prior probabilities, respectively) [6], or as a *hidden context* as described by Kuncheva [7]. The hidden contexts typically appear in incrementally acquired data that experience some type of perturbation, whether caused by noise, gradual or abrupt changes in the boundaries between classes, or even systematic trends that may be cyclical in nature. Hence, learning in a non-stationary environment is also known as *concept drift*, where concept refers to classes or class boundaries to be learned, and drift is the change in these boundaries. The drift can be gradual, abrupt, contracting, expanding or cyclical. When the drift is abrupt, the problem is more appropriately called *concept change* rather than *concept drift*. Both concept drift and concept change are evaluated through the experimental work described in this paper.

II. Commonly Used Approaches for Concept Drift

A. Windowing-Based Approaches

Earliest approaches to tracking a changing environment have sought to control how the data are used or learned by the algorithm through *windowing*. In windowing-based approaches, the data that fall into the most recent *time window* are thought to represent the currently valid information, which is then used to retrain a classifier. As new data become available, the window slides to include the most recent data. Previous data, now considered as irrelevant, are discarded. All information learned from such data is therefore forgotten. Variable length windows have been employed in algorithms such as FLORA [8] to handle various drift rates. A longer window is typically used for slowly varying environments, since more of the data is believed to be relevant at any given time. Conversely, a shorter window is used for fast changing environments, since a smaller segment of the data is then relevant to the current environment. Other data selection techniques have also been employed, which seek out instances that are believed to be most relevant to the current environment based on a comparison to previous instances [5;6].

B. Multiple Classifier Systems

Multiple Classifier Systems (MCS) (or ensemble) based algorithms represent a different approach to concept drift, and are particularly effective at providing a good balance between stability (retaining existing and relevant information) and plasticity (learning new knowledge). MCS-based approaches are characterized by an ensemble of classifiers which are combined to form a final decision. The challenge of MCS-based

[1] Manuscript received December 15, 2008. This work was supported by the National Science Foundation under Grant No: ECS 0239090.

Authors are with the Signal Processing and Pattern Recognition Laboratory, Electrical and Computer Engineering Department, Rowan University, Glassboro, NJ 08028 USA.

*Corresponding author: R. Polikar (E-mail: polikar@rowan.edu).

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

approaches is that of keeping the ensemble relevant to the current environment, which is often accomplished using some combination of voting techniques, batch or instance-based classifiers, and a forgetting mechanism.

Street's Streaming Ensemble Algorithm (SEA) is among the first MCS-based approaches, specifically developed for learning in non-stationary environments [9]. In SEA, an ensemble of classifiers is created for each consecutive time window (of predetermined size) of data. Classifiers are weighted according to a quality score, and their votes are combined to obtain the final decision. SEA employs an age-based forgetting mechanism, where old classifiers (that fall outside of the fixed ensemble size) are permanently deleted.

Ensemble weighting is introduced for drifting environments by Wang in [10;11] for large scale data-mining purposes using batch classifiers, which are trained on an entire window of data (hence not incremental learning). Classifier weights are related to their accuracy on the current training data and are calculated using mean square error. Both theoretical and empirical results indicate that this approach effectively gives more power to classifiers with low error in the current environment, and thus yields a performance substantially superior to that of an unweighted ensemble.

More recently, Dynamic Weighted Majority (DWM) is introduced by Kolter and Maloof [12;13], which uses an online learning approach, training and updating classifiers with each in-coming individual data instance. An error score is maintained for each expert in the ensemble, and experts are removed at a predetermined interval if their error exceeds some threshold. This algorithm also includes a re-training phase for each expert in order to keep the ensemble updated to the current environment.

Nishida's Adaptive Classifiers Ensemble (ACE) [14] takes a novel approach to MCS by combining both online and batch learning. Batch-trained classifiers are used to maintain prior knowledge, whereas an online classifier is updated on each new instance. The goal is to create a diverse ensemble that is competent in recent environments and also retain old knowledge. ACE employs a forgetting method for classifiers with high error, and also uses selective memory for making a final ensemble decision. The top performing classifiers (determined by a confidence interval) are polled while the rest are forgotten.

Perhaps one of the more interesting examples is Scholz and Klinkenberg's boosting based ensemble creation approach which maintains two ensembles – one trained on the currently available data, and one trained on a cache of previously seen data, and chooses the better of the two ensembles in each time step. Classifier weights are based on the LIFT of each classifier, which measures the correlation between the classifier's decision and the true class based on conditional probabilities. Changing values of LIFT for each classifier across time indicate the existence of drift [15]. The way in which the LIFT values are computed, however, restricts the algorithm to binary classification problems only.

C. Pruning

Many MCS-based algorithms have introduced some form ensemble pruning. Pruning has two primary objectives: first, to preserve memory and computation time in cases of long-term or large scale data-mining applications as suggested in [10;14;16]. The second objective is to maintain the ensemble's overall competency in the *current* environment as in SEA [9] as well as ACE [14]. Despite error-based weighted majority voting methods, assuring that classifiers are weighted with respect to their competence, an ensemble of classifiers may suffer from outvoting if the number of *in*competent classifiers (irrelevant to the current environment) becomes sufficiently large. Hence, a "forgetting" mechanism is often employed to enforce a limit on the ensemble size. When the limit is exceeded, a new classifier replaces an existing one, then believed to be irrelevant.

The most basic form of ensemble pruning is age-based pruning, also known as *replace-the-oldest*. A fixed ensemble size is chosen, and as new classifiers are generated, the oldest classifier that causes ensemble size to exceed the predetermined size is removed to make room for the newest classifier. Perhaps a more effective ensemble pruning approach, however, is error-based pruning, also known as *replace-the-loser* or *replace-the-weakest*. Here, an error-based criterion is imposed on all classifiers in the ensemble to determine – and remove – the least competent one(s) on the current environment. Competence can be relative to the mean squared error on current environment's training data (as in [8] and[14]) or some other quality criterion (as in [9]). One of the greatest concerns with any permanent ensemble pruning method, however, is the risk of forgetting information that can later become relevant again in recurring or cyclical environments. In such scenarios, ensemble members that are either old or have low accuracy in the current environment may very well be useful again if or when the distribution drifts *back* to an earlier state.

D. When Pruning is Necessary?

We have recently introduced an ensemble of classifiers approach, Learn^{++}.NSE, for incremental learning of concept drift in nonstationary environments [2;17]. Unlike other ensemble-based approaches, Learn^{++}.NSE does not discard any of the classifiers, but rather uses the classifiers' age-adjusted errors on current and past environments to determine if – or how much – each classifier should contribute to the final decision. Such an approach, of course, will only temporarily forget currently irrelevant information, but will be able to recall when such information becomes relevant again. On the other hand, this desirable property comes at a cost of increased complexity, as the algorithm will accumulate classifiers in perpetuity. It is therefore fair to ask whether – or when – a pruning mechanism can be beneficial, and if so, what kind of pruning mechanism is better suited for concept drift applications. To answer these questions, we have integrated and evaluated two controlled forgetting mechanisms for pruning old classifiers. Our overall goal is to test the effectiveness of the age-adjusted error-based weighting that is inherent in Learn^{++}.NSE in comparison to – as well as in combination to – the forgetting mechanism used by the two types of pruning methods, *replace-the-oldest* and *replace-the-weakest*. We also want to evaluate the tradeoffs (if such exists) between the usefulness of retaining old classifiers (e.g. in the case of recurring contexts) and the prevention of ensemble outvoting. Also provided is a cross-comparison of base classifiers using MLP, SVM, and Naïve Bayes classifiers.

III. LEARN^{++}.NSE

A. Overview

As an MCS-based algorithm, Learn^{++}.NSE expects to receive data in sequential batches, where each batch of data represents a snapshot of the current form of the distribution, which is expected to change over time. More specifically, Learn^{++}.NSE is provided with a series of training datasets $\{\boldsymbol{x}_i^t \in X; y_i^t \in Y\}, i = 1, \cdots, m^t$, where $\boldsymbol{x}_i^t$ is the i^{th} instance obtained from the t^{th} dataset (environment), drawn from an unknown distribution $P^t(\boldsymbol{x}, y)$, the current form of a possibly changing distribution at time t. At time $t+1$, we obtain a new training dataset drawn from $P^{t+1}(\boldsymbol{x}, y)$. At each time step, there may or may not have been a change in the environment, and if there were, the rate of this change is not known, nor assumed constant. Furthermore, we presume previously seen datasets – whether any of them is still relevant or not – are no longer available. Hence, the algorithm needs to work in an incremental fashion, and any information previously provided by earlier data must necessarily be stored in the parameters of the previously generated classifiers. Most existing concept drift algorithms, particularly those that use a windowing approach, do not make this restriction, and hence cannot be considered as incremental learning algorithms.

Learn^{++}.NSE then trains a new classifier with each dataset that is received, and adds this classifier to the ensemble. As the ensemble grows over time with addition of new classifiers, the potential for outvoting from irrelevant classifiers also increases. Thus, Learn^{++}.NSE employs a unique dynamic weighting mechanism that allows all classifiers (regardless of age) that are relevant to the current environment to contribute to the final decision with a voting weight that is proportional to its training data performance on the current dataset. This is accomplished by using an age-adjusted error that evaluates each classifier's performance at current and all previous environments. Performances on current and recent past receive a higher weight. Classifiers with high error are then *temporarily* ignored. This strategy justifies the retention of old classifiers, especially in cases of recurring contexts and cyclical data.

Two pruning methods are integrated into Learn^{++}.NSE and compared to the original algorithm that retains all classifiers. The first pruning approach is a time-based strategy, *replace-the-oldest*, in which the oldest classifier is replaced by an incoming classifier trained on the current environment to maintain a fixed ensemble size. The second approach is weight-based pruning, *replace-the-weakest*, where the classifier with the lowest accuracy on the current environment is dropped to make room for a new classifier. The weighting and voting strategies are the same as those used for a normal (un-pruned) ensemble in Learn^{++}.NSE. The algorithm is described in the following paragraphs, whose detailed pseudocode appears Figure 1.

B. Algorithm Description

Given the current training dataset, $\mathfrak{D}^t$, the primary free parameter of Learn^{++}.NSE is the selection of the supervised classification algorithm to be used as the BaseClassifier. This is the classification algorithm using which all classifiers of the ensemble are trained. Three such algorithms have been evaluated in this work, namely, the naïve Bayes (NB - an online learning algorithm), multilayer perceptron (MLP) and the support vector machine (SVM), the latter two of which are batch learning algorithms. We later show that Learn^{++}.NSE is largely invariant to the choice of the BaseClassifier.

The training data of size m^t is drawn from the current distribution $P^t(\boldsymbol{x}, y)$ at time t. As mentioned earlier, the current environment or distribution may have drifted in some way from the prior distribution $P^{t-1}(\boldsymbol{x}, y)$. At each time step, Learn^{++}.NSE initializes a distribution over each sample in the current dataset. Before training, the distribution is updated based on the error of the existing ensemble evaluated on the new batch of training data (that is, the current environment), providing a scalar measure on how much the current ensemble, i.e., the composite hypothesis H^t, already knows the data from the new environment. The normalized distribution then assigns a higher weight to instances $\boldsymbol{x}_i$ that are incorrectly classified under the hypothesis H^t.

Input: For each dataset $\mathfrak{D}^t$ $t = 1,2, \ldots$
Training data $\{\boldsymbol{x}_i^t \in X; y_i^t \in Y = \{1, \ldots, c\}\}, i = 1, \cdots, m^t$.
Supervised learning algorithm **BaseClassifier**
Ensemble size s
Do for $t = 1,2, \ldots$

If $t = 1$, **Initialize** $D^1(i) = w^1(i) = 1/m^1$, $\forall i$, (1)
Go to step 3.
Endif

1. Compute error of the existing ensemble on new data
$E^t = \sum_{i=1}^{m^t} (1/m^t) \cdot [\![H^{t-1}(x_i) \neq y_i]\!]$ (2)
2. Update and normalize instance weights
$w_i^t = \frac{1}{m^t} \cdot \begin{cases} E^t, H^{t-1}(x_i) = y_i \\ 1, otherwise \end{cases}$ (3)
Set $D^t = w^t / \sum_{i=1}^{m^t} w_i^t \implies D^t$ is a distribution (4)
3. Call **BaseClassifier** with $\mathfrak{D}^t$, obtain $h_t: X \to Y$
4. Evaluate all existing classifiers on new data $\mathfrak{D}^t$
$\varepsilon_k^t = \sum_{i=1}^{m^t} D^t(i) \cdot [\![h_k(x_i) \neq y_i]\!]$ for $k = 1, \ldots, t$ (5)
If $\varepsilon_{k=t}^t > 1/2$, generate a new h_t.
If $\varepsilon_{k<t}^t > 1/2$, set $\varepsilon_k^t = 1/2$,
$\beta_k^t = \varepsilon_k^t / (1 - \varepsilon_k^t)$, for $k = 1, \ldots, t$ (6)
5. Compute the weighted average of all normalized errors for k^{th} classifier h_k: For $a, b \in \mathbb{R}$
$\omega_k^t = 1/\left(1 + e^{-a(t-k-b)}\right)$, $\omega_k^t = \omega_k^t / \sum_{j=0}^{t-k} \omega_k^{t-j}$ (7)
$\bar{\beta}_k^t = \sum_{j=0}^{t-k} \omega_k^{t-j} \beta_k^{t-j}$, for $k = 1, \ldots, t$ (8)
6. Ensemble Pruning: **If** $t > s$
   a. Age-based: Remove h_{t-s} from the ensemble
   b. Error-based: Remove h_k where $\varepsilon_k^t = \max_{k=1 \ldots t} \varepsilon_k^t$
Endif
7. Calculate classifier voting weights
$W_k^t = \log(1/\bar{\beta}_k^t)$, $for\ k = 1, \ldots, t$ (9)
8. Obtain the final hypothesis
$H^t(x_i) = \arg\max_c \sum_k W_k^t \cdot [\![h_k(x_i) = c]\!]$ (10)

Fig. 1. Learn^{++}.NSE algorithm

Such an instance weighing scheme effectively focuses the algorithm on the previously misclassified instances on which the individual classifiers' new error is evaluated. Since such misclassified instances are likely to come from the current (and possibly previously unseen parts of the) environment, such an approach allows the error-based voting weights to focus on the current environment.

Once the new classifier is trained (Step 3), the error ε_k^t of each classifier h_k is evaluated based on its performance on the current training data $\mathfrak{D}^t$ (Step 4). Note that the distribution D^t itself is used to calculate the error in order to give more credit to classifiers which are accurate over previously misclassified instances (Equation 5). The individual error of each classifier is later used for calculating its voting weights. If the newly trained classifier's error on the current environment exceeds ½, that classifier is discarded and a new one is trained; if the error of a previously generated classifier exceeds ½, however, its weight is set to ½. An error of ½ yields a normalized error β_i^t of 1 (Equation 6) at the current time step, which carries zero voting weight (Equation 9).

Prior to calculating classifier weights, the normalized error is weighted using a non-linear sigmoid function to give more preference to each classifier's performance in the recent environment(s). Final "age-adjusted error averaged" voting weight of each classifier h_k at time t is then the logarithm of $1/\bar{\beta}_i^t$.

Ensemble pruning, which introduces the cap on ensemble size, s, as an additional free parameter, occurs in Step 6. Age-based pruning checks the current ensemble size, and permanently removes the oldest classifier (and its weight and prior error information) if the size exceeds this cap, s. Alternatively, error-based pruning selects the classifier with the highest error on the current training data to be permanently removed from the ensemble, provided that the ensemble size has reached the cap. All remaining classifiers are then combined through weighted majority weighting using their age-based error adjusted weights.

IV. Experimental Results

Several datasets simulating different scenarios of nonstationary environments, such as abrupt vs. gradual vs. cyclical drift, have been generated to yield some insight into the comparison of Learn^{++}.NSE with different pruning variations.

The following structure is used in all simulations: experiments begin at $t = 0$ and end at some arbitrary time $t = 1$. Within this interval, a total of T consecutive batches of data are presented for training, where each batch is assumed to be drawn from a drifting environment. Thus, the number T determines the number of time steps, or snapshots taken from the data throughout the period of drift. A large T corresponds to a low rate of drift seen by the algorithm, whereas a small T corresponds to a high effective drift rate, since the algorithm sees fewer snapshots of the data over time. In these experiments, the number of snapshots is kept the same for each data set, and we focus on the comparison of pruning methods. An analysis of the original Learn^{++}.NSE using various amounts of effective drift rate can be seen in [2]. As one would expect, the ability of the algorithm to track the changing environment is inversely proportional to the rate of drift; the slower the change, the better the tracking.

A. Triangular Drift Data

The triangular drift problem is a three-class synthetic data set ruled by a Gaussian distribution, which experiences drift in mean along the three edges of a triangle. Three classes undergo a rotational drift in a triangular pattern. Figure 2 shows the location of each class distribution at times $t = 0$, $t = 1/6$, $t = 1/3$, and $t = 1/2$, along with the direction of drift. The entire experiment is comprised of *two* complete rotations of each class along the path. Random noise is added to the experiment in order to prevent the appearance of identical snapshots from a recurring distribution. The parametric equations which govern these paths are shown in Table 1. The number of time steps chosen for this test was $T = 200$. At each time step, we select a total of 20 samples from each of the three class distributions, which serve as the current training data $\mathfrak{D}^t$.

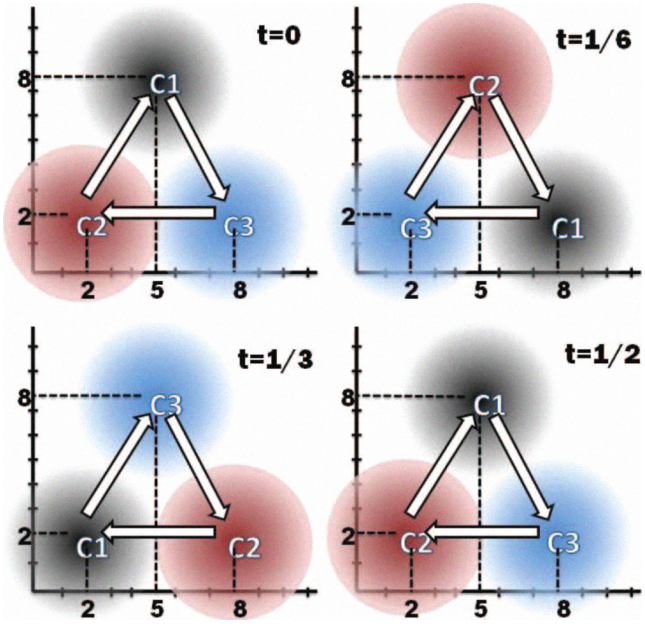

Fig. 2. Four snapshots from the path of triangular drift

Table 1. Parametric equations governing path of drift

	$0 < t < 1/6$ & $1/2 < t < 2/3$				$1/6 < t < 1/3$ & $2/3 < t < 5/6$			
	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y
C1	5+9t	8-18t	2	2	8-18t	2	2	2
C2	2+9t	2+18t	2	2	5+9t	8-18t	2	2
C3	8-18t	2	2	2	2+9t	2+18t	2	2

	$1/3 < t < 1/2$ & $5/6 < t < 1$			
	μ_x	μ_y	σ_x	σ_y
C1	2+9t	2+18t	2	2
C2	8-18t	2	2	2
C3	5+9t	8-18t	2	2

The results given in Figure 3, for each of the NB, MLP and SVM used as the BaseClassifer, shows the maximum achievable performance using a Bayes Classifier (since the distribution is known), the Learn^{++}.NSE algorithm, the Learn^{++}.NSE with two pruning implementations (age-based and error-based), and the performance of a single classifier trained only on the current training data. We note that this is an important benchmark, since the single classifier never carries the extra baggage of what then may be irrelevant data.

Each individual performance curve is color coded and enclosed with similarly colored background shading, denoting the 95% confidence interval over 50 independent trials. The wider the shading, the larger is the confidence interval (please see the electronic version for best visualization of the color coded results in these figures). Ensemble size cap was set as 25 in all experiments, as used in previously cited works.

Our focus here is not necessarily on overall performance (although the algorithm seem to track the Bayes classifier – plotted in orange - quite well); but rather to make inferences based on the usefulness of pruning methods at various stages in the process. We make the following observations from the results in Figure 3. Age-based pruning (in green) is clearly inadequate, which is often outperformed by a single classifier (red). This makes sense, as old data are not always irrelevant, particularly in recurring contexts as the one featured here.

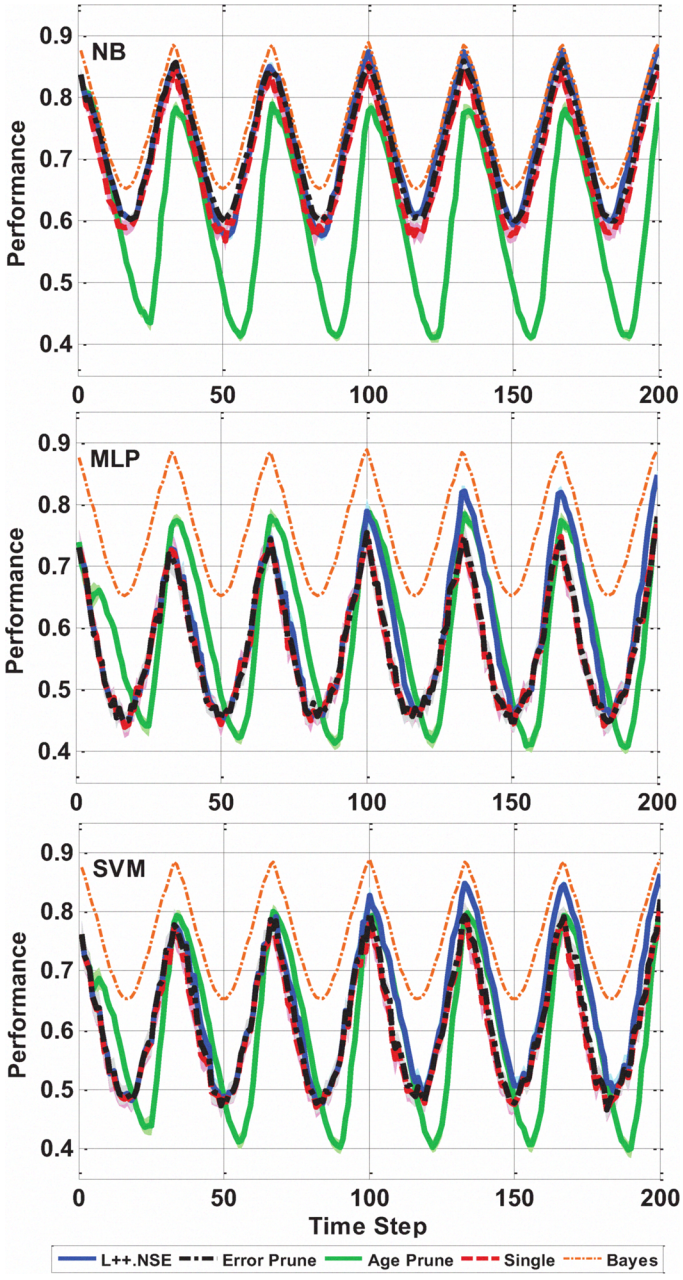

Fig 3. Comparative results on the three-class triangular drift data

Conversely, the original Learn^{++}.NSE (in blue) *and* the error-based pruning (in black) always perform as good as, or better than the single classifier. The error-based pruning never significantly outperforms a single classifier, while the original Learn++.NSE *does* show a statistical improvement during the second cycle/rotation of the data. We attribute the overall similarity of these methods to the relatively simple nature of the classification problem. Note that the sinusoidal characteristic of the performance curves are also due to symmetric and rotating nature of the distributions. Finally, we also observe that the algorithm Learn^{++}.NSE shows very similar behavior with all of the base classifiers, indicating that Learn^{++}.NSE itself is mostly independent of the base classifier being used.

B. Rotating Checkerboard Data

The second experiment is a unique, non-Gaussian data set which resembles a rotating checkerboard. As shown in Figure 4, the rotation makes this deceptively simple looking problem particularly challenging, as the decision boundaries change rather drastically at each time step. Figure 4 shows half of an entire rotation ($\alpha = 0$ to π), indexed to the parameter α, where the axis of rotation is the lower left corner of the board. Note that after half a rotation, data are drawn from a recurring environment, as the $[\pi\ 2\pi]$ interval will create an identical distribution drift to that of the $[0\ \pi]$ interval. As before, random noise is introduced to prevent identical training snapshots from appearing. Each training dataset was kept particularly small, comprising merely 25 samples (total from both classes) to add another layer of small-data-set challenge to this problem.

Figure 5 shows the performance result of 50 independent trials on test data comprised of 2601 points (a 51 by 51 grid) at each time step (the snapshots shown in Figure 4 are in fact the test data at those time instances). Once again, the comparison is made between Learn^{++}.NSE, two pruning methods, and a single classifier using three different base classifiers, the naïve Bayes, the MLP and the SVM. The Bayes classifier benchmark is not available in this scenario, since the dataset is not Gaussian. As before, each performance curve is enclosed by its 95% confidence interval to determine statistical significance of the performance differences.

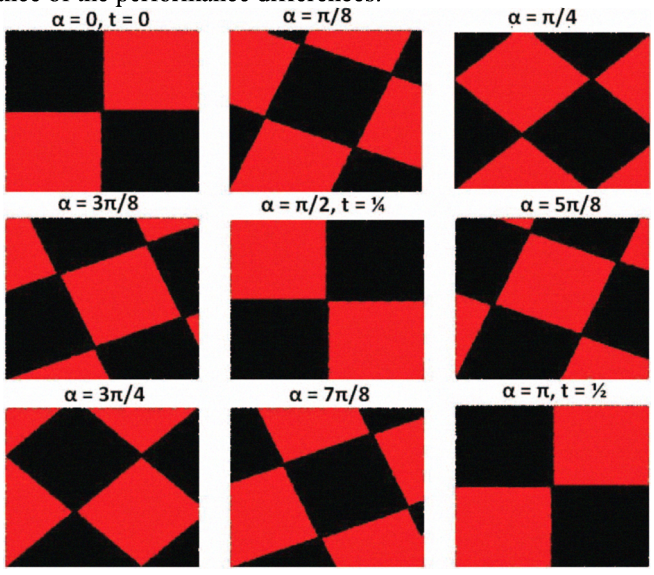

Fig. 4. Sample snapshots of the rotating checkerboard problem.

We make the following observations from the results in Figure 5. First and foremost, we note that the original Learn^{++}.NSE algorithm, which retains all of the classifiers, now performs better than all other algorithms, including both versions of itself that features a pruning mechanism. Furthermore, the differences are statistically significant at all times for NB, and most time steps for MLP and SVM (please see color figures in the electronic version of this document).

More specifically, Learn^{++}.NSE outperforms the single classifier as well as age-based pruning version in all cases and at all times during the experiment. Using MLP and SVM as base model, the error-based pruning approach also outperforms the age-based method and the single classifier (yet without statistical significance), but never outperforms Learn^{++}.NSE.

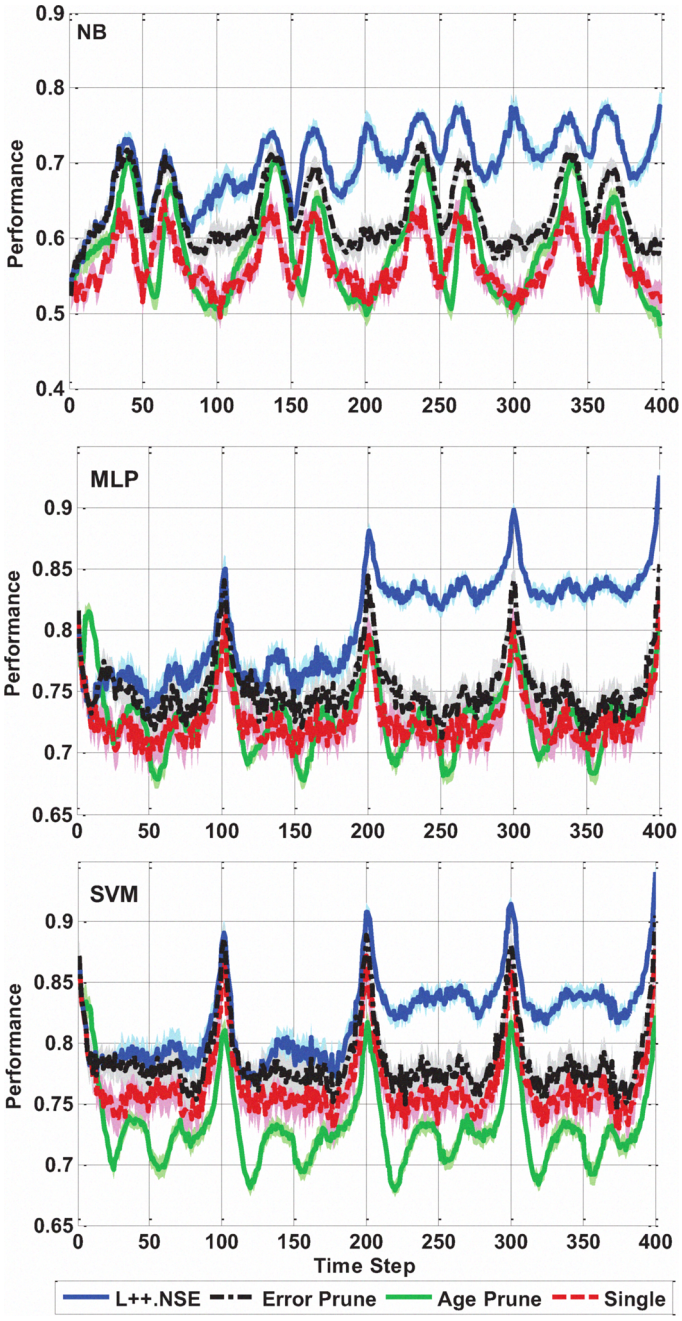

Fig 5. Comparative results on the two-class rotating checkerboard data.

Furthermore, Learn^{++}.NSE increases this margin between itself and error-based pruning during the second half of the rotation of the checkerboard, where the distribution repeats its first rotation drift. This makes sense, because during the second half of the rotation Learn^{++}.NSE can take advantage of all of the classifiers that were generated during the first half rotation. The age-based pruning has no access to those classifiers, whereas error-based pruning has only partial access to those classifiers to the extent allowed by its fixed ensemble size (25, in this case).

Also of interest are the periodic spikes of performances for all base classifier types and for all versions of the algorithms. We note that these spikes occur in predictive and regular intervals of 100 time steps which correspond to 90 degree rotations of the board. At this angle, the decision boundaries are substantially simpler, and the spikes in the performances merely correspond to classification problem being simpler at these time instances.

C. *SEA Concepts*

The third experiment involves an environment which undergoes both steady periods of no drift as well as sharp and abrupt changes in the class boundaries. This dataset was developed by Street et al., discussed in [9], and has been used by other proposed algorithms as a benchmark. The SEA (streaming ensemble algorithm) dataset contains two classes of three dimensional data (three features), only two of which are relevant to the classification problem. Hence the third dimension represents noise. The class labels are determined by using the sum of the first two features of the data, such that class 1 is assigned to all data whose sum is greater than some threshold, and class 2 is assigned to the rest. That is, for instance i, class label is $c_i = 1$ if $f_{i,1} + f_{i,2} \geq \theta_t$, and $c_i = 2$ otherwise. This effectively separates the classes by a two-dimensional hyper-plane.

Drift occurs as a result of a change in the value of the threshold θ_t, which corresponds to an abrupt change in the class labels. The threshold θ controls the changing concepts. As before, experiment begins at $t = 0$ and completes at $t = 1$; and as in [9], we use $\theta_t = 8$ for $t = 0$ to 0.25; 9 for $t = 0.25$ to 0.5; 7 for $t = 0.5$ to 0.75, and 9.5 for $t = 0.75$ to 1. The changes are introduced at evenly spaced intervals throughout the test, with each change being more drastic than the last. We have also added a 10% class noise to the training data as in [9].

We also follow the training data sizes introduced in [9], where a total of 50,000 total points are introduced as training data (25,000 points per class), and 250 points are introduced at each time step, corresponding to a total of 200 time steps in the experiment from t=0 to t=1.

We note that this experiment characterizes two different scenarios of i) sharp changes between environments; and ii) convergence during a stationary environment, as opposed to gradual drift shown in the previous two experiments. The results are shown in Figure 6, again for each of the three BaseClassifiers mentioned above.

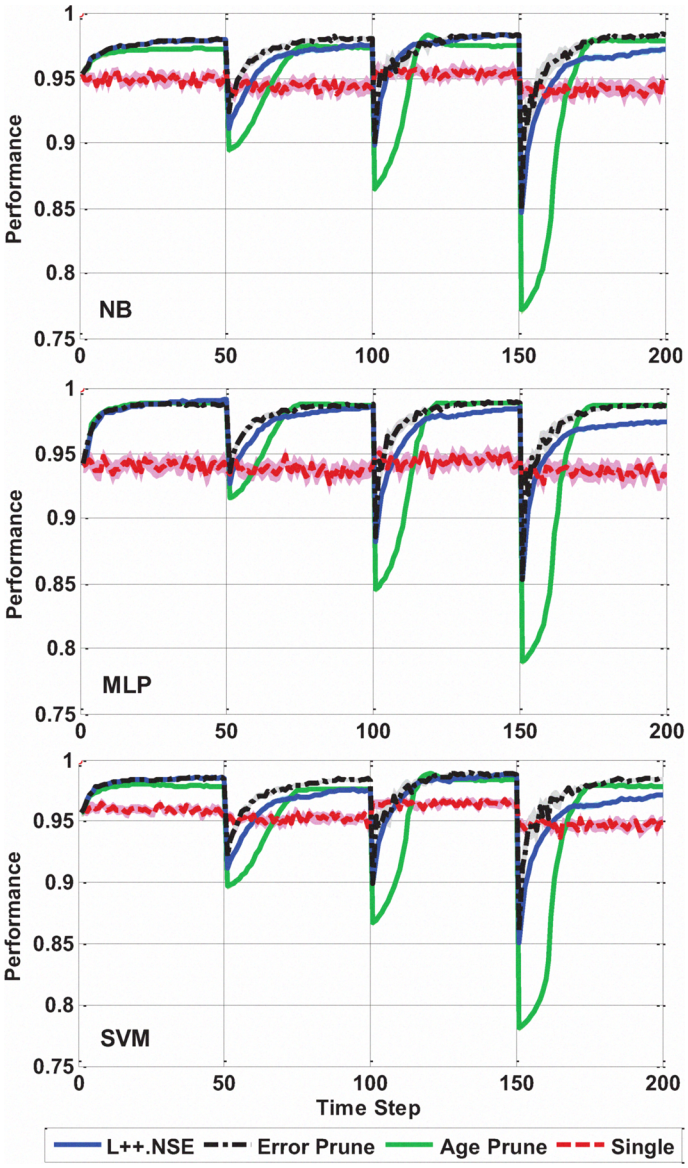

Fig. 6. SEA Concept data performance with 95% confidence interval

The figure of merit in this scenario is not the specific classification performance of the algorithm, but rather how quickly the algorithm can recover from an abrupt change. So the SEA data represent a concept change problem, rather than a concept drift problem. We observe that the single classifier has a fairly consistent performance throughout the experiment. This makes sense: unlike the rotating checkerboard problem, where the characteristics of the decision boundaries changes in time, the nature of the decision boundary in the SEA dataset, a (linear) line with identical slope, remains unchanged in all four cases. Furthermore, unlike the previous cases where the decision boundaries changed gradually, there is no explicit need to retain any of the previous classifiers. Since there is no extra baggage from older classifiers, single classifier does not experience any steep performance drop. Regardless of the base classifier used, the single classifier performance remains around 95%.

Such is not the case, however, for the ensemble-based algorithms, which need to properly reweigh the old classifiers that are currently in the ensemble, each time new data arrive. During those stretches where there is no concept drift, all ensemble-based algorithms behave more or less similarly, and gradually increase their classification performance to reach an asymptotic value, which is a well established feature of the ensemble-based algorithms. When a sudden concept change occurs, the decision boundaries generated by the previous classifiers become (mostly) incorrect, with only one (the most recent) classifier generating the correct decision boundary. The combination of large number of misclassifications then results in the steep drop in performance as seen with all ensemble algorithms and all base classifier configurations.

The second observation we note is that – despite the steep drop in performance – all ensemble-based algorithms are able to recover from the drop, and in fact exceed the performance of the single classifier with statistically significant margins (as before, the shading around each curve represents that classifier's 95% confidence interval). Among the three ensemble-based approaches, however, there are significant differences in recovery time. The age-based pruning approach performs most sluggishly after a concept change, as it takes much longer for this approach to achieve its asymptotic value. Both the original Learn^{++}.NSE as well as it error-based pruning version, on the other hand, recover much faster, the error-based pruning approach being the faster of the two. This also makes sense, since after the concept change, there is relatively little reason to retain the old classifiers, and the error-based pruning gets rid of all old classifier in at most s steps, where s is the number of classifiers to be retained in the ensemble. We should also add that the difference in performance as well as the recovery time between the original Learn^{++}.NSE and its error-based pruning version is not consistently statistically significant, as seen in Figure 6. This indicates that the error based – and age adjusted - weighting allows Learn^{++}.NSE to effectively identify and ignore the irrelevant classifiers.

Finally, as in all previous cases, we note that the Learn^{++}.NSE and both of its versions that feature a pruning mechanism behave very similarly with identical trends regardless of the base classifier being used. This is also noteworthy because such invariance allows the user to choose the classification algorithm that is most appropriate for the application at hand.

V. Conclusions & Discussions

A primary concern with learning in nonstationary environments is how to handle previously acquired knowledge that may or may not be currently relevant. Algorithms that use a windowing approach over the incoming instances assume that any data that falls outside of the current window is irrelevant and hence train a single classifier only with the data that falls into the current data window. Such approaches never have to worry about the issue of how to handle previously acquired knowledge, as all such knowledge is discarded every time a new classifier is trained. On the other hand, these approaches risk a significant performance drop if and when the previous knowledge is still – at least partially – relevant. Ensemble-based approaches overcome this difficulty by retaining the previously generated knowledge in the ensemble members. However this solution comes at a cost of increased computational and memory resources due to maintaining and reweight-

ing each classifier with each addition to the ensemble. Furthermore, retaining an ensemble also makes the algorithm less agile to track fast changing environments. One way to alleviate these problems is to use a pruning strategy. Two most commonly used pruning strategies in ensemble-based systems are age-based and error-based pruning, both of which retain a fixed size ensemble. Our previously introduced Learn^{++}.NSE algorithm does not discard any of the classifiers and instead relies on a strategic dynamic weight update rule. The question of our interest in this effort was therefore when is it advantageous to retain all classifiers and rely only on the weight update rule to ignore irrelevant decisions of old classifiers and when is it advantageous to use a pruning strategy.

The experiments conducted in this study give several clues to the behavior of ensemble-based algorithms for nonstationary environments. First, ensemble approaches outperform a single classifier, even when the concept change / drift is rapid or abrupt as in the SEA dataset. Interestingly, we see that if the old knowledge has little or no relevance to the current environment, it appears that retaining all classifier members as in Learn^{++}.NSE can still provide an advantage (though often not significant) but it never causes any performance degradation. Pruning-based approaches may nevertheless be preferred due to their lower computational cost and lower likelihood of overfitting. Similar arguments can also be made for gradual / slowly changing environments, so long as such environments do not experience a cyclical change. If a cyclical change is expected or even suspected, then all classifiers should be retained as demonstrated by the performance of the original Learn^{++}.NSE with the rotating checkerboard problem.

The choice of the pruning strategy appears to be a less difficult decision: error-based pruning should, in general, be preferred over the age-based pruning. Limiting the ensemble size according to lowest error tends to increase the ensemble's ability to react to any environment change, but specifically to sharp changes, as seen in the SEA experiment. We observed that the error-based pruning can react to such sharp changes much faster than the age-based pruning.

Finally, we should add Learn^{++}.NSE and its pruning based versions are likely to be most beneficial in learning in nonstationary environments i) if the environment provides so little data at each time step that a new classifier trained with such limited data cannot sufficiently model the current environment; and/or ii) if the previously generated classifiers retain at least partial currently relevant information. We note that cyclic environments necessarily satisfy the second condition.

Our future work will include comparison of Learn^{++}.NSE to other approaches of concept-drift learning, such as the streaming ensemble algorithm (SEA) itself, and the dynamic weighted majority on these simulated data as well as real-world data as such data become available.

REFERENCES

[1] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, no. 1, pp. 17-61, 1988.

[2] M. Karnick, M. Ahiskali, M. D. Muhlbaier, and R. Polikar, "Learning concept drift in nonstationary environments using an ensemble of classifiers based approach," *World Congress on Computational Intelligence, International Joint Conference on Neural Networks*, 2008, pp. 3455-3462.

[3] D. P. Helmbold and P. M. Long, "Tracking drifting concepts by minimizing disagreements," *Machine Learning*, vol. 14, no. 1, pp. 27-45, 1994.

[4] A. Kuh and T. Petsche, "Learning Time Varying Concepts with Applications to Pattern Recognition Problems," *Signals, Systems and Computers, 1990. 1990 Conference Record Twenty-Fourth Asilomar Conference on*,T. Petsche, Ed. vol. 2, 1990, p. 971.

[5] J. C. Schlimmer and R. H. Granger, "Incremental Learning from Noisy Data," *Machine Learning*, vol. 1, no. 3, pp. 317-354, Sept.1986.

[6] A. Tsymbal, "Technical Report: The problem of concept drift: definitions and related work," Trinity College, Dublin, Ireland,TCD-CS-2004-15, 2004.

[7] L. I. Kuncheva, "Classifier Ensembles for Changing Environments," *Multiple Classifier Systems (MCS 2004)*, Lecture Notes in Computer Science , vol. 3077, 2004, pp. 1-15.

[8] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69-101, 1996.

[9] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," *Seventh ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-01)*, 2001, pp. 377-382.

[10] H. Wang, W. Fan, P. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 226-235.

[11] P. Wang, H. Wang, X. Wu, W. Wang, and B. Shi, "A Low-Granularity Classifier for Data Streams with Concept Drifts and Biased Class Distribution," *IEEE Transactions on Knowledge and Data Engineering,* vol. 19, no. 9, pp. 1202-1213, 2007.

[12] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: a new ensemble method for tracking concept drift," *3rd IEEE Int. Conf. on Data Mining (ICDM 2003)*, 2003, pp. 123-130.

[13] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: an ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755-2790, 2007.

[14] K. Nishida, K. Nishida, and K. Yamauchi, "Adaptive Classifiers-Ensemble System for Tracking Concept Drift," *Machine Learning and Cybernetics, 2007 International Conference on*,K. Yamauchi, Ed. vol. 6, 2007, pp. 3607-3612.

[15] M. Scholz and R. Klinkenberg, "Boosting Classifiers for Drifting Concepts," *Intelligent Data Analysis, Special Issue on Knowledge Discovery from Data Streams*, vol. 11, no. 1, pp. 3-28, 2007.

[16] C. Fang, W. Yizhou, and C. Zaniolo, "An adaptive learning approach for noisy data streams," *Fourth IEEE International Conference on Data Mining (ICDM 2004)*, 2004, pp. 351-354.

[17] M. Karnick, M. D. Muhlbaier, and R. Polikar, "Incremental Learning in Non-Stationary Environments with Concept Drift Using a Multiple Classifier Based Approach," *International Conference on Pattern Recognition (ICPR 2008)*, 2008.