Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

Semi-supervised Learning in Nonstationary Environments

Gregory Ditzler and Robi Polikar

***Abstract*—Learning in nonstationary environments, also called learning concept drift, has been receiving increasing attention due to increasingly large number of applications that generate data with drifting distributions. These applications are usually associated with streaming data, either online or in batches, and concept drift algorithms are trained to detect and track the drifting concepts. While concept drift itself is a significantly more complex problem than the traditional machine learning paradigm of data coming from a fixed distribution, the problem is further complicated when obtaining labeled data is expensive, and training must rely, in part, on unlabelled data. Independently from concept drift research, semi-supervised approaches have been developed for learning from (limited) labeled and (abundant) unlabeled data; however, such approaches have been largely absent in concept drift literature. In this contribution, we describe an ensemble of classifiers based approach that takes advantage of both labeled and unlabeled data in addressing concept drift: available labeled data are used to generate classifiers, whose voting weights are determined based on the distances between Gaussian mixture model components trained on both labeled and unlabeled data in a drifting environment.**

***Index Terms*—concept drift; non-stationary environments; unlabeled data; ensemble systems; incremental learning**

I. Introduction

NONSTATIONARY environments are typically associated with streaming data whose underlying distribution changes over time. When such a change occurs in data, the outcome is referred to as concept drift, which has been receiving increasing attention from the machine learning community [1,2]. Such an attention is not unfounded: the traditional assumption made by most machine learning algorithms – that the data come from a fixed distribution – does not hold in many real world applications, such as analysis of long-term financial, climate, or electricity demand data. Hence, many of the traditional classification algorithms are simply not equipped to handle concept drift. Considering the streaming nature of data generated by nonstationary environments, using traditional machine learning algorithms on such data inevitably causes subpar performance on future datasets. Learning in nonstationary environments is therefore an incremental learning problem, which is associated with two conflicting objectives: retaining previously learned knowledge that is still relevant, and replacing any obsolete knowledge with current information. These conflicting objectives are collectively known as the *stability-plasticity dilemma* [3]. The problem is further complicated however, if the discarded information later becomes relevant again, a not-so-uncommon phenomenon observed in so-called cyclical environments (such as climate or any seasonal data). In such cases, the ability to *recall* previous knowledge would be useful. Therefore, algorithms designed to learn concept drift must be capable of adaptively adjusting its structure or parameters to accurately track a variety of drifting environments, such as slow, fast, gradual, abrupt or cyclical changes in the environment.

G. Ditzler and R. Polikar are with the Signal Processing and Pattern Recognition Laboratory in the Electrical & Computer Engineering Department at Rowan University located in Glassboro, NJ, 08028, USA. Author email: gregory.ditzler@gmail.com, and polikar@rowan.edu.

This work is based on material supported by the National Science Foundation under Grant No: ECCS-0926159.

Since incremental learning in nonstationary environments requires a streaming source of data (either online or in batches) to learn the new concept, a further difficulty arises if obtaining labeled data is expensive, or the labels do not become available at the time data are acquired. This is because the typical scenario in streaming nonstationary data applications requires the algorithm to be trained with data available at time step t, which is then evaluated on data that become available at time step $t+1$. In applications where labeling the data is a time consuming process (compared to data acquisition), labels from the previous batch may not be available at the time new data become available.

As unlabeled data are often less expensive or easier to obtain, a number of algorithms – so called semi-supervised and transductive approaches – have been developed, which can make use of (possibly limited) labeled and (usually abundant) unlabeled data [4,5]. As stated in [6], inductive semi-supervised approaches use labeled data $\{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ and unlabeled data $\{\boldsymbol{x}_k\}_{k=l+1}^{l+u}$ to learn a function $f: \mathcal{X} \rightarrow \mathcal{Y}$ such that f is a good predictor on future unlabeled data beyond $\{\boldsymbol{x}_k\}_{k=l+1}^{l+u}$. Transductive approaches, on the other hand, use labeled data $\{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ and unlabeled data $\{\boldsymbol{x}_k\}_{k=l+1}^{l+u}$ to learn a function $f: \mathcal{X}^{l+u} \rightarrow \mathcal{Y}^{l+u}$ such that f is a good predictor on $\{\boldsymbol{x}_k\}_{k=l+1}^{l+u}$. In a transductive learning setting, f is only defined for a given training sample and is not expected to predict outside $\{\boldsymbol{x}_k\}_{k=l+1}^{l+u}$. Several researchers have shown that strategic use of unlabeled data could result in better generalization performances compared to using a strictly supervised algorithm with the limited labeled data only. It is reasonable to expect that a semi-supervised approach can improve the ability of a classifier to track nonstationary environments that provide limited labeled and abundant unlabeled data. The labeled data are available for training and the unlabeled data are provided for testing; however, the training and testing data may be sampled from different sources. Yet, developing semi-supervised and transductive algorithms for concept drift applications have been mostly underexplored.

In this contribution, an incremental learning algorithm is presented that is designed to use unlabeled data drawn from a distribution different from previously seen distributions to update the weights of an ensemble of classifiers. The approach integrates Gaussian mixture models for determining classifier-voting weights.

The rest of this paper is organized as follows: Section II presents background literature into clustering and concept drift, Section III provides a motivation for the proposed approach, Section IV describes the current approach in detail, Section V contains the results of our preliminary experiments, and Section VI provides discussions, concluding remarks and future efforts.

II. Background

Concept drift algorithms can be categorized in several ways, such as online vs. batch algorithms – based on the number of instances used for each update; active vs. passive approaches – based on whether an explicit drift detection mechanism is used or drift is simply assumed; as well as single classifier vs. multiple classifier systems (MCS) based approaches. MCS-based approaches have been widely used for a wide spectrum of data mining problems including concept drift, learning new classes [7], data fusion [8], and feature selection [9]. Earliest attempts of addressing concept drift include single classifier, passive batch algorithms STAGGER [10] and FLORA [11], which use a sliding window to choose a block of (new) instances to train a new classifier. The window size is modified via *window adjustment heuristic*, based on how fast the environment is changing, making the algorithm at least partially "active". FLORA has a built-in forgetting mechanism with the implicit assumption that those instances that fall outside of the window are no longer relevant, and the information carried by them can be forgotten. On the other hand, some active drift detection approaches for concept drift include CUSUM (cumulative sum) and confidence interval intersection based approaches that only take corrective action when drift is detected [12,13,14]. These are also examples of single classifier based approaches.

Ensemble based approaches constitute a different group of concept drift algorithms. Ensemble approaches have been widely popular because of their simple, yet well-founded theory based on variance reduction, and their ability to strike a meaningful balance between stability and plasticity, thus avoiding *catastrophic forgetting* associated with other algorithms that replace the existing classifier with a new one trained on the new data only [15]. Street and Kim's *streaming ensemble algorithm* (SEA) was one of the earliest ensemble based techniques designed for nonstationary environments [16]. SEA simply adds new classifiers to the ensemble trained on the new data as new data become available; however uses an upper limit on the ensemble size (typically 25). To do so, SEA prunes additional classifiers (in excess of 25) by determining the quality of each classifier, measured by its correct classification performance with respect to the entire ensemble's performance. Kolter & Maloof present *dynamic weighted majority* (DWM), an on-line learning ensemble designed for concept drift applications [17]. DWM maintains a pool of classifiers and weighs these classifiers in a semi-heuristic method that reduces classifier weights when instances are misclassified. Classifiers with low weights are then discarded. More recently, Bifet et al. presented a new method of on-line bagging, which uses *adaptive size Hoeffding trees* (ASHT) [18]. ASHT is motivated by the idea that smaller trees adapt more quickly to a changing concept than a large tree, but a large tree will perform well over periods where the concepts are not changing. In our previous work, we introduced Learn^{++}.NSE, a member of Learn^{++} family of incremental learning algorithms [19], which also generates a classifier with each new datasets, but uses a dynamic weighting strategy that uses a weighted sum of each classifier's error over current, recent and old environments [20]. The weighted sum is a sigmoid applied to the classifier's errors, which weighs recent errors more heavily than errors on older environments. Such a weighting scheme allows a classifier that performs well in recent times to be rewarded with a higher weight, regardless of its age.

Other concept drift algorithms combine the ensemble approach with the sliding window to hold instances for later use either for training or as a change estimator/detector. One such example is `ADWIN`, an adaptive sliding window approach for drifting data streams [21]. Whenever two large enough sub-windows exhibit "distinct enough" expectations, `ADWIN` determines the corresponding expected values are significantly different and drops the older portion of the sub-window.

There has also been an increasing interest in clustering algorithms for concept drift, and a software package has recently been released geared towards clustering and classification with concept drift present in data streams [22]. StreamKM++ is a data stream clustering algorithm that computes a small weighted sample of the data stream and uses a modified k-means to select the initial values in the cluster [23]. However, the modified k-means requires access to the training data, which is not feasible for data stream mining. As a work-around, the authors present a non-uniform sampling approach. Similarly, *D-stream* is a density based clustering framework for evolving data streams where little prior knowledge is known about the data [24]. Rather than retain old data as some clustering algorithms do, D-stream partitions the feature space into discretized grids and maps new instances to the corresponding grid. Grids are clustered based on their density. *ClusTree* is another data stream clustering algorithm capable of detecting concept drift, novelty and outliers in the stream [25]. The clustering algorithm uses an adaptive index structure for maintaining summaries concerning the data stream. Finally, *Den-stream* uses "dense" micro-clusters to summarize clusters with arbitrary shape [26]. With the exception of a few recent works, a semi-supervised approach that can take advantage of abundant unlabeled as well as limited labeled data has been mostly underexplored, despite parallel set of works of classification and clustering in concept drift. The existing works include primarily those of Zhang et al [27,28]. In one approach they use relational k-means and a semi-supervised SVM to work with labeled and unlabeled data with concept drift in [27]. In another approach, the same authors ensemble algorithms for combining classifiers and clusters for learning from data streams [28].

III. Determining Classifier Weights

Consider the problem of estimating a set of Bayes-optimal discriminate functions based on the outputs of the classifiers in an ensemble [29]. The discriminate function can be formed by a weighted combination of the classifier outputs. However, we wish to determine a non-heuristic, optimal set of weights for member classifiers. The Bayes-optimal discriminate function for class ω_j on instance $\boldsymbol{x}$ is given by (1), where $\mathbf{h}$ is a vector of labels predicted by T classifiers in the ensemble. Assuming all classifiers in the ensemble provide independent outputs, we may apply the naïve Bayes rule on the conditional probability to formulate (2) which can be rewritten as in (3).

$$g_j(\boldsymbol{x}) = \log\{P(\omega_j)p(\mathbf{h}|\omega_j)\} \tag{1}$$

$$= \log\left\{P(\omega_j)\prod_{k=1}^{T} p(h_k|\omega_j)\right\} \tag{2}$$

$$= \log P(\omega_j) + \sum_{k:h_k=\omega_j} \log\frac{1-\varepsilon_k}{\varepsilon_k} + \sum_{k=1}^{T}\varepsilon_k \tag{3}$$

where

$$\varepsilon_k = 1 - p(h_k|\omega_j) \tag{4}$$

is the error of the k^{th} classifier hypothesis h_k in predicting the true label ω_j. The last term in (3) has no dependence on ω_j, and therefore it can be dropped from the discriminate. The discriminate function is further reduced if one can reasonably assume all classes have equal prior probabilities. Thus, the weights for the classifiers in the ensemble can be assigned using (5).

$$W_k \propto \log\frac{1-\varepsilon_k}{\varepsilon_k} \tag{5}$$

Equation (5) states that the optimal weights for each classifier in an ensemble can be determined if the error of the classifier can be compute on a static dataset. The error of the k^{th} classifier, ε_k, is typically estimated by computing the error on a validation dataset. In order to compute the weights, there must be a significant amount of labeled data to reliably estimate the error. The objective of this work is to use unlabeled data, possibly drawn from a different distribution (i.e., different from the distribution that generated the training data) to aid in assigning a weight to each classifier. Let S_k represent the source distribution whose data was used to generate a classifier h_k at time stamp k where $k = 1,2,\dots,t$ and t is the most recent time stamp for which training data is available. If the field data (i.e., test data) is sampled from a different source distribution, $S_{t+t_\beta} \neq S_t$ where $t + t_\beta$ is an arbitrary time stamp in the future, the weights $W_k^{(t)}$ computed based on the errors of the classifiers trained on S_t will not be optimal. Using the weights given by $W_k^{(t+t_\beta)}$ (as shown in (6)) would be ideal for testing our ensemble on data sampled from S_{t+t_β}; however, there is not labeled data at time stamp $t + t_\beta$ to estimate $\varepsilon_k^{(t+t_\beta)}$. The problem is, then, how to estimate $W_k^{(t+t_\beta)}$ when (field) data is not yet labeled.

$$W_k^{(t+t_\beta)} \propto \log\frac{1-\varepsilon_k^{(t+t_\beta)}}{\varepsilon_k^{(t+t_\beta)}} \tag{6}$$

A clustering algorithm is used to represent the distribution of the unlabeled data drawn from a source distribution S_{t+t_β}. Specifically, a mixture of Gaussians with K components is generated from the unlabeled data, and a separate mixture of Gaussians for each class in the labeled data set sampled from source S_t. Assuming the drift in the data is gradual or incremental, one can try to determine a correlation between the two sets of mixture models. This processed is described in detail in the next section.

IV. Approach

The proposed *weight estimation algorithm* (WEA) is a batch based incremental learning algorithm for concept drift that receives batches of labeled and unlabeled data at each time stamp. However, in line with the typical streaming data applications, the unlabeled data are not available at the time of training, but only become available at the time of testing. Specifically, at time t a batch of labeled data is used to train a new classifier. Upon completion of the training, there is unlabeled test data, which is used to adjust classifier voting weights. Furthermore, again in line with the assumption of possibly continuous drift, we accept that the source distributions that generated the training data and the test data may be different, and that access to the labels of the field data is not available until after they are evaluated by the ensemble. Using weights determined from the labeled data, per (5), may result in suboptimal raw classification accuracy if there is a significant amount of bias between the distributions of the labeled and unlabeled data sources. Thus, the goal is to use the unlabeled data and a set of mixtures models formed on the previous data sources to estimate the weights for all the classifiers in the ensemble *before* we begin to classify the unlabeled field data. Only after the field data have been classified, the true labels of the instances are received so that the accuracy of the ensemble can be computed.

WEA assumes there is limited concept drift in the incremental learning scenario. By limited concept drift, the assumption is that the drift is not completely random; rather there is a structure to the drift. Thus, the underlying source generating the data is evolving with time and not randomly selecting new sources to generate data. The sources generating the labeled and unlabeled data presented at a future time stamp are not radically different. Instead, the learning scenario experiences a gradual or incremental drift. After all, a source changing randomly between time stamps cannot be learned. Therefore, we formalize our limited drift assumption as follows: the Bhattacharyya distance between a known (labeled) component and its future position (unlabeled at the time of testing) must be less than the Bhattacharyya distance between the known component and every other future (unlabeled) component from a different

```
Input: Labeled training data $\mathcal{D}^{(t)} = \{\boldsymbol{x}_i \in \mathcal{X}; y_i \in \mathcal{Y}\}$
        where $i = 1, \dots, m^{(t)}$
    Unlabelled field data $\mathcal{B}^{(t)} = \{\boldsymbol{x}_j \in \hat{\mathcal{X}}\}$
        where $i = 1, \dots, n^{(t)}$
    $K_c$: number of centers for the $c$th class in a GMM
    $q^{(t)}$: number of instances generated to estimate the
          classifier error
    BaseClassifier learning algorithm

for $t = 1,2, \dots$ do
  1. Call BaseClassifier on $\mathcal{D}^{(t)}$ to generate $h_t: X \to \mathcal{Y}$
  2. Generate a GMM for each class with $K_c$ centers in
     labeled $\mathcal{D}^{(t)}$. Refer to these mixture models as $\mathcal{M}_c^{(t)}$.
  3. Generate a GMM with $\sum K_c$ centers from unlabeled
     $\mathcal{B}^{(t)}$. Refer to this mixture model as $\mathcal{N}^{(t)}$.
  4. Compute Bhattacharyya distance between the
     components in $\mathcal{N}^{(t)}$ and the components in $\mathcal{M}_c^{(t)}$.
     Assign each component in $\mathcal{N}^{(t)}$ with the label of the
     closest component in $\mathcal{M}_c^{(t)}$. Refer to this mixture as
     $\mathcal{N}_c^{(t)}$
  5. Generate synthetic data from $\mathcal{N}_c^{(t)}$ and compute the
     error for each classifier on synthetic data
         $\hat{\varepsilon}_k^{(t)} = \frac{1}{q^{(t)}} \sum_{l=1}^{q^{(t)}} [\![ h_k(\boldsymbol{x}_l) = y_l ]\!]$
     where $q^{(t)}$ is the number of synthetic instances
     generated and $k = 1,2, \dots, t$
     if $\hat{\varepsilon}_k^{(t)} > 1/2$ then
       $\hat{\varepsilon}_k^{(t)} = 1/2$
     end if
  6. Compute classifier voting weights for the field data
         $W_k^{(t)} \propto \log \frac{1 - \hat{\varepsilon}_k^{(t)}}{\hat{\varepsilon}_k^{(t)}}$
  7. Classify the field data in $\mathcal{B}^{(t)}$
     $H^{(t)}(\boldsymbol{x}_j \in \mathcal{B}^{(t)}) = \arg\max_{c \in \Omega} \sum_{k=1}^{t} W_k^{(t)} [\![ h_k(\boldsymbol{x}_j) = y_j ]\!]$
end for
```
Fig. 1. Weight Estimation Algorithm (WEA) pseudo code

class. This assumption simply means that the drift of *mixture components* is gradual and not radical.

Let $\mathcal{M}_c^{(t)}(i)$ be the ith component of a mixture model at time stamp t for class c, which contains K_c components and $i, j, k = 1, \dots, K_c$. Assuming Gaussian mixture models (GMM) are used in WEA, the limited drift assumption is given by (7), which states that the Bhattacharyya distance between the ith component of the cth class in $\mathcal{M}_c^{(t)}$ and the jth component of the cth class in $\mathcal{M}_c^{(t+1)}$ must be less than the distance between ith component of the cth class in $\mathcal{M}_c^{(t)}$ and kth components of the c'th class in $\mathcal{M}_{c'}^{(t+1)}$ where $c \neq c'$. We will see why this assumption is made and what

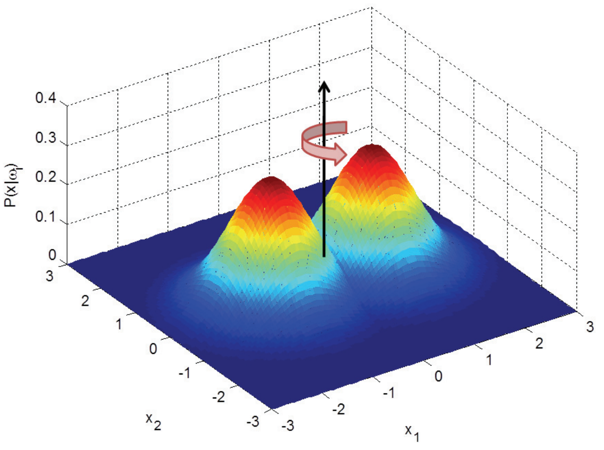

Fig. 2. Initial likelihood of the rotating Gaussian mixture dataset. Each mixture in the concept drift problem represents a class. The mixtures drift in a clockwise rotation around on another.

are the possible outcomes of a violation of this assumption. Note that the limited drift assumption requires that (7) should hold for all i, j, k with $c \neq c'$ for all classes.

$$\delta_B\left(\mathcal{M}_c^{(t)}(i), \mathcal{M}_c^{(t+1)}(j)\right) < \delta_B\left(\mathcal{M}_c^{(t)}(i), \mathcal{M}_{c'}^{(t+1)}(k)\right) \qquad (7)$$

WEA, whose pseudo code is given Fig. 1, works iteratively as new data become available, as follows: At time t WEA is presented with a batch of labeled training data $\mathcal{D}^{(t)}$ and a batch of unlabeled field data $\mathcal{B}^{(t)}$. The source distribution that generated the data in $\mathcal{B}^{(t)}$ may be different from that of $\mathcal{D}^{(t)}$ due to concept drift in the data. A new classifier is generated using a supervised base classifier algorithm on $\mathcal{D}^{(t)}$. Because this is an incremental learning algorithm, only the current training data may be used for training at any time, that is, prior data is considered unavailable. The base classifier is not a weak learning algorithm as used in AdaBoost and many other ensemble based approaches. Rather, a classifier that is a good predictor is generated and not a “rule of thumb”. Next, WEA generates a Gaussian mixture model (GMM) for each class c (where c is any arbitrary class), with K_c centers. K_c should be chosen – possibly based on prior knowledge or experience – that the GMM will be able to reasonably approximate the underlying data distribution. In this work Expectation-Maximization (EM) initialized by K-means is used to form the mixture of Gaussians. The GMM for class c, obtained from the labeled data at time t is referred to as $\mathcal{M}_c^{(t)}$. The mixture model $\mathcal{M}_c^{(t)}$, for all classes, consists of a total of K components, K_c of which represent class c. Another Gaussian mixture model is generated from the unlabeled data in $\mathcal{B}^{(t)}$ when such data arrive. A GMM is generated from the data in $\mathcal{B}^{(t)}$ with K components where K is given by (8)

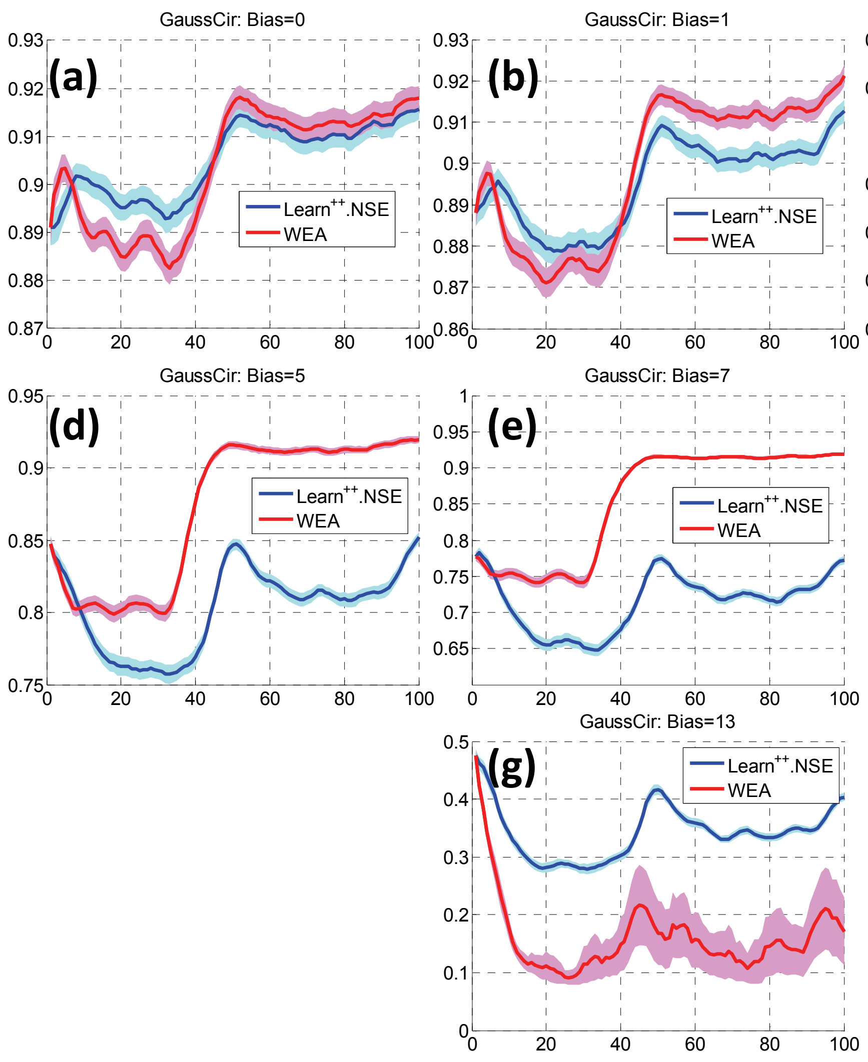

Fig. 3. Raw classification accuracy comparison of WEA and Learn^{++}.NSE evaluated on the *GaussCir* dataset. The experiment is run over 200 time stamps and batch sizes of 1000 with approximately equal prior probabilities. The level of bias between the labeled and unlabeled data varies as follows: (a) zero bias, (b) 1 time stamp, (c) 3 time stamps, (d) 5 time stamps, (e) 7 time stamps, (g) 10 time stamps, and (g) 13 time stamps.

$$K = \sum_{c \in \Omega} K_c \tag{8}$$

where the set Ω consists of all classes in the incremental learning problem. The mixture model obtained from the unlabeled data is referred to as $\mathcal{N}^{(t)}$. Thus, $\mathcal{N}^{(t)}$ is the model of the data sampled from an unknown source, which is an evolution of the previous (known) model, $\mathcal{M}_c^{(t)}$. Since labels are not available for $\mathcal{B}^{(t)}$, we do not know which mixture components belongs to which class in $\mathcal{N}^{(t)}$. Next, WEA seeks to determine where the components in $\mathcal{M}_c^{(t)}$ have drifted to in $\mathcal{N}^{(t)}$ by measuring the similarity between the components in both models. To do this, the Bhattacharyya distances are computed, given by (9), between a component in $\mathcal{N}^{(t)}$ and all components in $\mathcal{M}_c^{(t)}$. The component in $\mathcal{N}^{(t)}$ is assigned to the label of the mixture in $\mathcal{M}_c^{(t)}$ with the smallest Bhattacharyya distance. Thus, we now observe why the assumption of limited drift was stated because if (7) does not hold a component will be temporarily mislabeled.

$$\delta_B = \frac{1}{8}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \left(\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right)^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \frac{1}{2}\log\left(\frac{\left|\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}\right) \tag{9}$$

The components of $\mathcal{N}^{(t)}$ that are closest to the component $\mathcal{M}_c^{(t)}$are assigned the label c because of the minimum distance. We then call the GMM obtained from these components $\mathcal{N}_c^{(t)}$. Note that $\mathcal{N}_c^{(t)}$ simply represents a (temporarily) labeled component of $\mathcal{N}^{(t)}$. Once the (estimated) labels are assigned to the components in $\mathcal{N}^{(t)}$, WEA generates a batch of synthetic data with class associations by sampling from the GMMs in $\mathcal{N}_c^{(t)}$. The

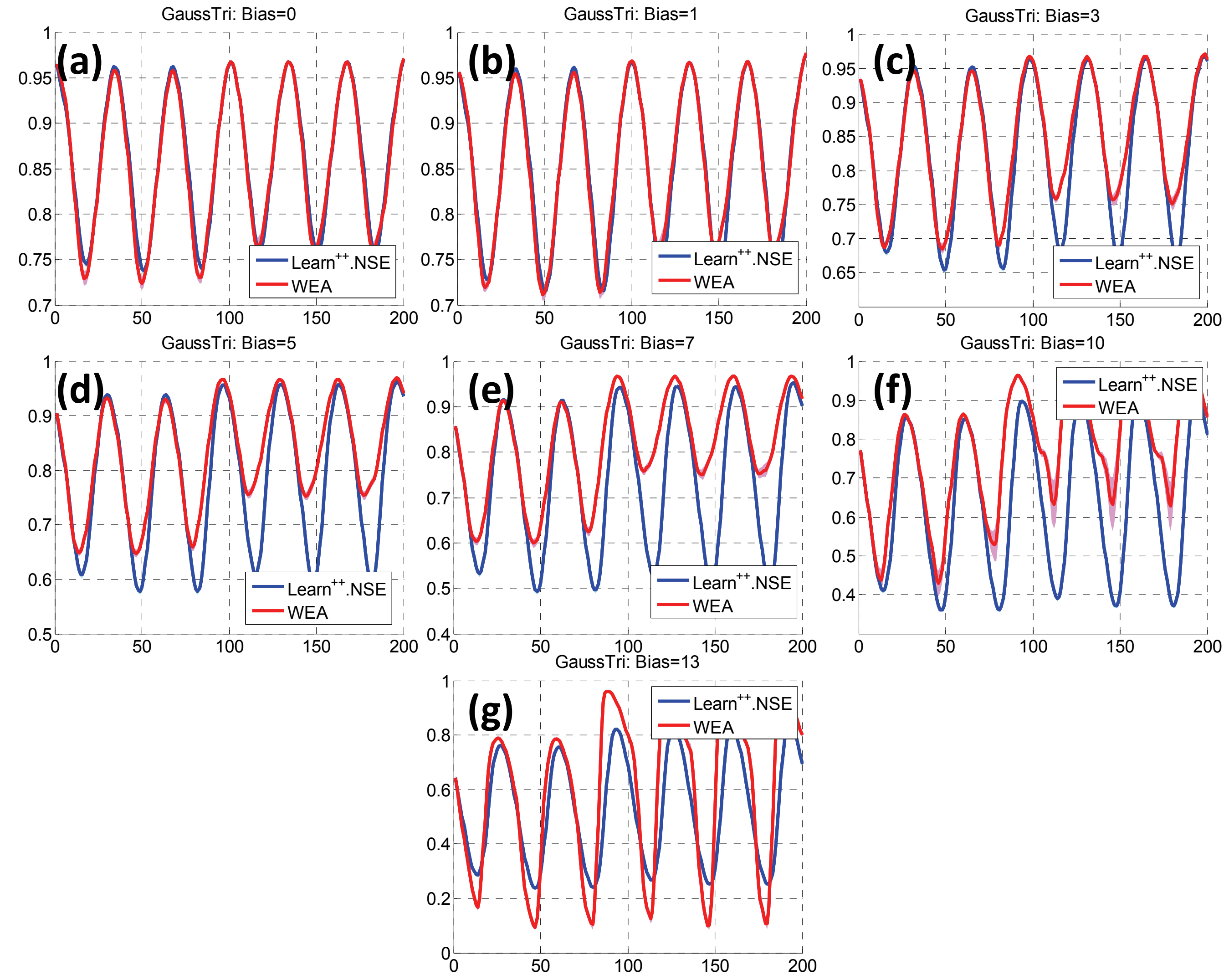

Fig. 4. Raw classification accuracy comparison of WEA and Learn^{++}.NSE evaluated on the *GaussTri* dataset. The experiment is run over 200 time stamps and batch sizes of 1000 with approximately equal prior probabilities. The level of bias between the labeled and unlabeled data varies as follows: (a) zero bias, (b) 1 time stamp, (c) 3 time stamps, (d) 5 time stamps, (e) 7 time stamps, (g) 10 time stamps, and (g) 13 time stamps.

sample size, $q^{(t)}$, is a free parameter of the algorithm. If the aforementioned assumption has not been violated, the data sampled from $\mathcal{N}_c^{(t)}$ should have a distribution similar to that of $\mathcal{B}^{(t)}$, but now with estimated class labels. The synthetic data, along with its assigned (temporary) labels are used to estimate the expected error, denoted by $\hat{\varepsilon}_k^{(t)}$, on $\mathcal{B}^{(t)}$. The expected error of the classifiers on the synthetic data will reasonably estimate the error on $\mathcal{B}^{(t)}$, if the GMMs accurately represent the data in $\mathcal{D}^{(t)}$ and the limited drift assumption is held. Thus, the weights computed by WEA should approach the original weights we were seeking to estimate in (6). The weights of the classifiers, $W_k^{(t)}$, are inversely proportional to the logarithm of the estimated average error on $\mathcal{B}^{(t)}$. Finally, the unlabeled data are classified using a weighted majority vote, where the final class labels are determined.

V. Experimental Results

A series of synthetic datasets were used to demonstrate the capabilities of the proposed approach for using unlabeled data in tracking nonstationary environments. We also compare this algorithm to Learn^{++}.NSE whose pseudo code can be found in [20]. Learn^{++}.NSE uses a similar ensemble approach and weighted majority voting to track drift; however Learn^{++}.NSE only uses labeled data. Using identical base classifiers, this comparison then allows us to determine whether the proposed GMM based approach can use unlabeled data to better track nonstationary environments. Synthetic datasets are used to strictly control the distributions of the data and the rate/type of drift present in the incremental learning problem. Each experiment is averaged with 50 independent trials. The lightly shaded regions around each plot represent the 95% confidence interval that allows us to test for statistical significance. A

CART decision tree was used as the base classifier both for the WEA and Learn^{++}.NSE.

A. Datasets used for Experimentation

The rotating Gaussian dataset is comprised of two Gaussian components, each for a different class, rotating around one another. The class means are determined by using the parametric equations $\boldsymbol{\mu}_1^{(t)} = [\cos\theta_t, \sin\theta_t]^T$, $\boldsymbol{\mu}_2^{(t)} = -\boldsymbol{\mu}_1^{(t)}$, $\theta_t = \frac{2\pi m}{N}t$, with fixed class covariance matrices given by $\Sigma_1 = \Sigma_2 = 0.5 \times \mathbf{I}$, where m is the number of cycles, t is an integer valued time stamp ($t = 0,1,\dots,N-1$) and $\mathbf{I}$ is a 2×2 identity matrix. The experiment is run over 2 cycles with 1000 train/test instances in each batch, and a varying level of bias between the training and testing datasets. Bias was injected by sampling the testing dataset from a future time stamp. This dataset is referred to as *GaussCir*.

A second Gaussian dataset was created, consisting of three components, each belonging to a different class moving in a triangular pattern. The environment experiences reoccurring concepts for a total of two rotations. The parametric equations that govern the mean and standard deviations for the x and y components are presented in Table I. Each batch of training/testing data contains 1000 instances. The drift rate for the triangular Gaussian dataset as well as the circular Gaussian dataset remains constant throughout the duration of the experiment. This dataset is referred to as *GaussTri*.

B. Results

Fig. 3 shows the performance of WEA and Learn^{++}.NSE evaluated on the *GaussCir* dataset under varying levels of bias between the training and testing datasets. WEA uses 1-mixture component for each class. WEA performs on par Learn^{++}.NSE when there is no bias between the training/testing batches. Both algorithms have a significant boost in performance when a reoccurring environment is encountered, which happens at time step 50. WEA maintains nearly the same performance as it did without any bias when the bias is increases. However, Learn^{++}.NSE's performance begins to drop off rapidly as the bias increases. The effect of bias on Learn^{++}.NSE can be observed with small amount of bias as shown in Fig. 3(b) and 3(c), referring to a bias of 1 and 3 time stamps, respectively. This result is expected because Learn^{++}.NSE computes its classifier weights from a batch of data that was drawn from a significantly different distribution than the distribution used for the evaluation of Learn^{++}.NSE, and the weights of Learn^{++}.NSE are not updated with respect to the test data. Many other concept drift algorithms may suffer the same drop in performance because information in the unlabeled data is not used to adjust classifier voting weights. WEA maintains a dominant performance over Learn^{++}.NSE until the bias in the data becomes large (when bias=13 time steps). Note that since this is a relatively easy problem (2 classes with one mixture each); violating the limited drift assumption becomes quite detrimental to the classification performance whereas a problem with a large number of mixture components may not experience the same degradation. Whether this is indeed the case is currently been tested and will be reported in future correspondences.

TABLE I. PARAMETRIC EQUATIONS CONTROLLING THE TRIANGULAR DRIFT SCENARIO OVER THE INTERVAL OF $0 < t < 1$

	$0 < t < 1/6$, $1/2 < t < 2/3$				$1/6 < t < 2/6$, $2/3 < t < 5/6$			
	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y
ω_1	5+18t	8-36t	2	2	8-36t	2	2	2
ω_2	2+18t	2+36t	2	2	5+18t	8-36t	2	2
ω_3	8-36t	2	2	2	2+18t	2+36t	2	2

	$2/6 < t < 1/2, 5/6 < t < 1$			
	μ_x	μ_y	σ_x	σ_y
ω_1	2+18t	2+36t	2	2
ω_2	8-36t	2	2	2
ω_3	5+18t	8-36t	2	2

The preliminary results of WEA on the *GaussTri* dataset are shown in Fig. 4. The experiment is run with two cycles and each GMM uses 1-mixture component for each class, thus $K = 3$. Similar results are recorded for the *GaussTri* dataset as with *GaussCir*. WEA is a very strong predictor when the bias between the training and testing datasets is 3, 5, 7, or 10 time stamps. The response of WEA becomes slightly less stable when the bias increases further. However, WEA performs quite well when the bias between the labeled and unlabeled data is reasonable that is, when the limited drift assumption holds. WEA should perform well on a variety of problems where the distribution can be modeled with GMMs and the drift is gradual or incremental in nature. Gradual or incremental concept drift should imply that the limited drift assumption is held.

VI. CONCLUSION

We have presented a *weight estimation algorithm* (WEA) for determining classifier-voting weights when concept drift is present in incremental learning scenarios. WEA is an incremental ensemble based algorithm that uses labeled and to build classifiers and unlabeled data to aid in the calculation of the classifier voting weights before the data are classified. WEA was compared to Learn^{++}.NSE, an incremental learning algorithm for learning in non-stationary environments and empirical results indicate that WEA performs similarly to Learn^{++}.NSE when there is no bias between the labeled and unlabeled data. However, WEA showed significant improvement when bias was present between the distributions of labeled (training) and unlabeled (testing) batches. We have also demonstrated a potential weakness of our proposed WEA approach: clear violation of the limited drift assumption can result in poor overall accuracy compared to state of the art approaches like Learn^{++}.NSE. However, the violation of the limited drift assumption may not be as detrimental to the overall accuracy if the GMMs contain a large number of mixtures. This will be addressed in future research. The incremental learning scenarios presented in this paper offer a proof of concept for WEA and its application to concept drift and unlabeled data.

WEA can have potential issues when there is a uniform distribution throughout the entire experiment and the concept drift is induced via a changing decision boundary.

Therefore, WEA should be evaluated on larger sets of synthetic and real-world experiments, which constitutes part of our future work. Other work to investigate includes developing theoretical models to aid in the reliability and/or accuracy of a system that applies semi-supervised learning mechanisms to concept drift problems. Essentially, to what extent and under what conditions can unlabeled data be beneficial to aiding in classifying data that was sampled from sources different from ones presented for training?

REFERENCES

[1] L. I. Kuncheva, "Classifier ensembles for changing environments," in *Proceedings 5th Intl. Workshop on Multiple Classifier Systems*, 2004, pp. 1-15.

[2] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, vol. 9, pp. 56-68, 2008.

[3] S. Grossberg, "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks*, vol. 1, no. 1, pp. 17-61, 1988.

[4] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*. MIT Press, 2006.

[5] X. Zhu, "Semi-Supervised Learning Literature Survey," Computer Sciences, University of Wisconsin-Madison 1530, 2005.

[6] X. Zhu. (2009) Tutorial on Semi-Supervised Learning.

[7] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.NC: Combining Ensemble of Classifiers With Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 152-168, 2009.

[8] D. Parikh and R. Polikar, "An Ensemble-Based Incremental Learning Approach to Data Fusion," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 37, no. 2, pp. 437-450, 2007.

[9] P. Somol, J. Frim, and P. Pdil, "Criteria Ensembles in Feature Selection," in *Int'l Workshop on Multiple Classifier Systems*, 2009, pp. 304-313.

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

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

[12] C. Alippi and M. Roveri, "Just-in-time Adaptive Classifiers. Part I. Detecting non-stationary Changes," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145-1153, 2008.

[13] C. Alippi and M. Roveri, "Just-in-time Adaptive Classifiers. Part II. Designing the Classifier," *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2053-2064, 2008.

[14] C. Alippi, G. Boracchi, and M. Roveri, "Change Detection Tests Using the ICI rule," in *International Joint Conference on Neural Networks*, 2010, pp. 1190-1196.

[15] F. H. Hamker, "Life-long learning cell structures continuously learning without catastrophic forgetting," *Neural Networks*, vol. 14, no. 5, pp. 551-573, 2001.

[16] W. N. Street and Y. Kim, "A Streaming Ensemble Algorithm (SEA) for Large Scale Classification," in *ACM SIGKDD Intl. Conf. on Knowledge Discovery & Data Mining*, 2001, pp. 377-382.

[17] J. Z. Kolter and M. M. Maloof, "Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755-2790, 2007.

[18] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New Ensemble Methods For Evolving Data Streams," in *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2009.

[19] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31, no. 4, pp. 467-508, 2001.

[20] P. Muhlbaier and R. Polikar, "Multiple classifiers based incremental learning algorithm for learning nonstationary environments," in *IEEE Intl. Conf. on Machine Learning and Cybernetics*, 2007, pp. 3618-3623.

[21] A. Bifet and R. Gavalda, "Kalman filters and adaptive windows for learning in data streams," in *Intl. Conf. on Discovery Sicence*, 2006, pp. 29-40.

[22] A. Bifet, et al., "MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering," in *Workshop on Applications of Pattern Analysis*, 2010.

[23] M. Ackermann, et al., "StreamKM++: A Clustering Algorithms for Data Streams," in *SIAM ALENEX*, 2010.

[24] L. Tu and Y. Chen, "Stream Data Clustering Based on Grid Density and Attraction," *ACM Transactions on Computational Logic*, vol. 1, no. 1, pp. 1-26, 2008.

[25] P. Kranen, I. Assenty, C. Baldauf, and T. Seidl, "Self-Adaptive Anytime Stream Clustering," in *IEEE Intl. Conf. on Data Mining*, 2009, pp. 249-258.

[26] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-Based Clustering over an Evolving Data Stream with Noise," in *SIAM Conf. on Data Mining*, 2006, pp. 328-339.

[27] P. Zhang, X. Zhu, and L. Guo, "Mining Data Streams with Labeled and Unlabeled Training Examples," in *International Conference on Data Mining*, 2009, pp. 627-636.

[28] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and Cluster Ensembles for Mining Concept Drifting Data Streams," in *International Conference on Data Mining*, 2010, pp. 1175-1180.

[29] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, Inc., 2004.