Transductive Learning Algorithms for Nonstationary Environments

Gregory Ditzler, Gail Rosen and Robi Polikar

***Abstract*—Many traditional supervised machine learning approaches, either on-line or batch based, assume that data are sampled from a fixed yet unknown source distribution. Most incremental learning algorithms also make the same assumption, even though new data are presented over periods of time. Yet, many real-world problems are characterized by data whose distribution change over time, which implies that a classifier may no longer be reliable on future data, a problem commonly referred to as concept drift or learning in nonstationary environments. The issue is further complicated when the problem requires prediction from data obtained at a future time step, for which the labels are not yet available. In this work, we present a transductive learning methodology that uses probabilistic models to aid in computing ensemble classifier voting weights. Assuming the drift is limited in nature, the proposed approach exploits a probabilistic estimate to determine the class responsibility of components in a Gaussian mixture model (GMM), generated from labeled and unlabeled data. A general error bound is provided based on the ensemble decision, the probabilistic estimate of the GMM, and the true labeling function, which, unfortunately is never actually known.**

***Keywords*-concept drift; Gaussian mixture models; multiple classifier systems; unlabeled data**

I. Introduction

Concept drift is the problem of learning from a data source, whose distribution – intermittently or continuously – changes in time, and it is referred to as learning in *nonstationary environments*. The change in data structure can be abrupt or gradual, fast or slow, reoccurring, or otherwise. These drift types can lead to other forms of drifting environments (e.g., abrupt change in the evidence, $p(x)$, of a random variable x or the true decision function, $f(x)$). The drift in data can be *real* or *virtual* in nature. Real drift is a change in the conditional likelihood, $p(x|\omega)$ (the probability of x conditioned on class ω), while virtual drift is an observed change (e.g., change in class priors, $p(\omega)$) – as a result of an incomplete representation of the true distribution of the current data (e.g., due to sampling bias). Typically, we find that real and virtual drift occur at the same time, thus making the learning problem quite difficult [1]. Some researchers strictly focus on one scenario, such as virtual drift [2], while we try to address the problem more generally. The fundamental problem that concept drift presents is that classifiers created in the past become outdated or remain only partially relevant (e.g., some portions of the learning problem may not have changed or changed very little).

G. Ditzler and G. Rosen are with the Dept. of Electrical & Computer Engineering at Drexel University, Philadelphia, PA. They are supported by the National Science Foundation (NSF) CAREER award #0845827, NSF Award #1120622, and the Department of Energy Award #SC004335. Author email: gregory.ditzler@gmail.com, gailr@ece.drexel.edu

R. Polikar is with the Dept. of Electrical & Computer Engineering at Rowan University. He is supported by the NSF under Grant No: ECCS-0926159. Author email: polikar@rowan.edu

Consider, as an example, the problem of predicting precipitation from four features (average temperature, wind speed, humidity, and barometric pressure). Assume batch based processing where a classifier is generated on a season of data, e.g., spring, and must predict on data from the next season, summer, for which the labels are not yet available. There will likely be some change in $p(x)$ because of the temperature and barometric pressure changing from the seasonal transition, as well as a change in $p(x|\omega = \text{rain})$ and $P(\omega = \text{rain})$ as it may rain more in the spring than summer. Therefore, a classifier should be able to adapt to such a changing environment to predict on future data. On-line processing also faces similar problems: given a stream of instances, appearing one instance at a time, when should a classifier be modified or perhaps replaced by a new classifier, if $p(x)$, $p(x|\omega)$, or $P(\omega)$ change in time, possibly from one sample to the next.

Consider an adversary that controls the source from which the train/test sets are sampled. The adversary is therefore in charge of defining how the environment changes, as well as the rate of the change. Therefore, in classification problems where an adversary is in control of the drift, the data must be learnable even with the change. It is always possible for the adversary to select a source(s) (e.g., at random) that cannot possibly be predicted by the learning algorithm at each time stamp [3] (as true randomness cannot be learned). Therefore, and without any loss of generality, it is necessary to assume that the drift contains – at a minimum – some loose form of structure, and that it evolves over time. We will refer to this assumption as the *limited drift assumption*.

In this work, we first describe a multiple classifier systems (MCS) based approach to learn from batch data over time, where the sources generating the data are changing (or drifting). If the data are arriving in an online fashion, we can simply wait for a period to accumulate a batch of data prior to processing. The novelty of the approach is that we use *unlabeled (test) data* in a transductive learning setting to improve the performance of the ensemble in predicting labels to field data. Probabilistic models are generated on (labeled) training and (unlabeled) testing data. A maximum a posteriori (MAP) estimate is performed to infer the degree of responsibility each known class accepts to explain the parameters of the model generated with the unlabeled data. The probabilistic models are used to aid in the calculation of classifier voting weights. A loose error bound is produced for the MCS, written in terms of the ensemble hypothesis, the probabilistic model estimates, and the true labeling function. We also demonstrate the feasibility of a second, related, transductive learning algorithm that offers a less computationally complex solution to the problem than ensembles, under certain conditions.

This paper is organized as follows: Section II presents a brief overview of the related work on learning from concept drift with an emphasis on ensemble based systems, Section III presents our approach for learning concept drift with classifiers and Gaussian mixture models, Section IV presents results from synthetic and real-world experiments and finally, Section V discusses some final conclusions as well as future research directions.

II. Related Work

A variety of methods have been proposed to handle concept drift including single classifier methods, drift detection, and MCS based solutions to name a few. Typically, concept drift algorithms can be categorized as *active* or *passive* based approaches [4]. Active approaches seek to identify when change occurs in the data stream and then takes appropriate corrective action based on the amount and nature of the drift detected. A passive approach simply assumes that some amount of drift may be present in the data and adjusts algorithm parameters whenever new data become available. For example, a drift detection approach such as the the early drift detection method (EDDM), [5] and Hellinger distance drift detection method (HDDDM) [6], would be considered active, whereas dynamic weighted majority (DWM) [7], is a passive approach.

Many of the earliest approaches for concept drift use a single classifier and/or sliding window [3], [8], or use adaptive classifiers along with drift detection to learn in nonstationary environments. For example, Alippi et. al. propose an adaptive k-NN classifier with the computational intelligence cumulative sum (CI-CUSUM) for drift detection [9], [10] and more recently have presented classifiers for detecting gradual drift [11], [12], a more difficult problem than detecting an abrupt change. Another example of a single classifier approach is the uniFied Instance Selection algoritHm (FISH), which uses time and distance information to form training sets for a classifier [13]. Yamauchi proposes a single classifier design using radial basis function (RBF) neural networks, inspired by previous work for covariate shift; however, his approach relaxes the computational requirements of prior work [2].

Ensemble based approaches constitute a different group of concept drift algorithms that 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, ensemble based approaches can avoid catastrophic forgetting associated with other algorithms that replace the existing classifier with a new one trained on the new data only [14], [15]. In [16], ECSMiner is presented for learning concept drift and identifying novel classes in data streams. Kolter & Maloof present dynamic weighted majority (DWM), an on-line learning ensemble designed for concept drift applications [7]. 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. Another ensemble-based approach, Learn^{++}.NSE generates a classifier with each new dataset and uses a dynamic weighting strategy that features a weighted sum of each classifier's time-adjusted errors over current and recent environments [4]. More recently, Bifet et al. proposed an on-line bagging approach that uses adaptive size Hoeffding trees (ASHT) [17]. ASHT is motivated by the idea that smaller trees adapt more quickly to a changing concept than a large tree, but a large tree performs better over periods where the concepts are not changing. Also in [17], `ADWIN` bagging was presented, which combines the adaptive windowing algorithm, `ADWIN`, and bagging. Minku et. al. have proposed that diversity in classifier ensemble can also be informative for handling concept drift [18], [19].

Many of the batch-based learning algorithms update classifier parameters when labeled data are presented, then new data are classified [4], [20]. None of the aforementioned approaches have a mechanism to take advantage of unlabeled data, even though many applications provide test-then-train type data, where the labels of the test data are not immediately available. Climate data, financial data, power demand and pricing data are just a few examples of such applications. With the exception of a few works, such as that of Zhang et al. [21], which combines classification and clustering, taking advantage of unlabeled data has not been addressed within concept drift literature. More recently, we proposed a weight estimation algorithm (WEA) for updating classifier weights by relating Gaussian mixture models (generated at different time stamps on labeled and unlabeled data) through the Bhattacharyya distance [22].

In the next section, we describe a proof-of-concept methodology that exploits unlabeled field data to aid in updating the parameters of the ensemble. With the aforementioned limited drift assumption in place, we expect the ensemble to benefit from the information provided by the unlabeled data, and we present a loose error bound for the ensemble hypothesis. The approach presented in the next section is therefore a transductive learning algorithm for nonstationary environments.

III. Approach

The proposed approach, called TRansductive leArning eNSEmble (TRANSE) for concept drift, is designed to use knowledge learned from previous environments to infer class information about the target probability distribution that has no class information yet available (i.e., test data distribution). Thus, data from the target distribution are assumed to be unlabeled and sampled i.i.d. from an unknown distribution that has not been learned. TRANSE uses the unlabeled target data – sampled possibly from a different distribution than training distributions – to infer label information about each unlabeled data. This approach is considered transductive because the primary focus is on classification of the unlabeled data, not to make TRANSE a generalized prediction algorithm on distant future unlabeled data. That is because, a generalized prediction algorithm would not work well in situations were the source that generates the data changes with time.

Input: labeled data $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^m$, unlabeled data $\mathcal{B}_t = \{(x_q)\}_{q=1}^n$, **Learn** base learning algorithm, and $\mathbf{\Theta}$ GMM parameters

1: **for** $t = 1, 2, \ldots$ **do**
2: Call **Learn** with $\mathcal{D}_t$ and receive classifier (hypothesis) h_t
3: Compute GMM parameters $\mathbf{\Theta}_c^t$ for all classes $c = 1 \ldots, C$ from $\mathcal{D}_t$
4: Use $\mathcal{B}_t$ to compute GMM parameters $\mathbf{\Theta}_{\mathcal{B}}^t$
5: Compute $g_c(x)$ using (3) for $x \in \mathcal{B}_t$ (where $c = 1, \ldots, C$) and compute the expected error, $\epsilon_{l,t}$, for classifier l on $\mathcal{B}_t$, where $l = 1, \ldots, t$.
6: Update classifier voting weights

$$\alpha_{l,t} = \log \frac{1 - \epsilon_{l,t}}{\epsilon_{l,t}} \tag{1}$$

7: Predict class labels for $x_q \in \mathcal{B}_t$

$$H(x_q) = \arg\max_{y \in \mathcal{Y}} \sum_{l=1}^{t} \alpha_{l,t} [\![h_l(x_q) = y]\!] \tag{2}$$

8: **end for**
Output: Ensemble hypothesis $H(x_q)$ for $q = 1, \ldots, n$

Fig. 1. Transductive learning ensemble pseudo code (TRANSE)

TRANSE is designed to learn in scenarios where labeled data are available intermittently at some arbitrary time stamp t, which have been sampled from source $\mathcal{S}_t$. Some unlabeled data become available after training on the labeled data; however, the source used to sample the unlabeled data may be different from $\mathcal{S}_t$. Under the assumption of limited drift, TRANSE aims to transfer knowledge from $\mathcal{S}_t$ to infer possible class information about the unlabeled data. This transferred knowledge is then used to update an ensemble's parameters and classify the unlabeled data. We emphasize that samples drawn from a source at a particular time t are assumed to be i.i.d.

A. Algorithm Description

The pseudo code for TRANSE is shown in Fig. 1. A density estimation algorithm is needed by TRANSE, for which we choose Gaussian mixture models (GMM). TRANSE is provided with labeled training data $\mathcal{D}_t$ with data pairs (x_i, y_i) for $i = 1, \ldots, m_t$ at time stamp t. Unlabeled field (target) data, $\mathcal{B}_t$, is presented after the training data have been processed. The unlabeled dataset, $\mathcal{B}_t$, is not only the target data to be classified by TRANSE, but it is also used by TRANSE to update its model parameters - hence the transductive learning component of TRANSE.

The algorithm begins by calling **Learn**, any supervised classification algorithm, with the labeled data, $\mathcal{D}_t$, to generate a new classifier (Line 2). Then for each class c in the training dataset $\mathcal{D}_t$, a GMM is generated, where $\mathbf{\Theta}_c^t$ represents the GMM parameters (i.e., mean & covariance components) for class c at time stamp t (Line 3). There are K_c components in $\mathbf{\Theta}_c^t$ and the total number of components for all classes is

Input: labeled data $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^m$, unlabeled data $\mathcal{B}_t = \{(x_q)\}_{q=1}^n$, and $\mathbf{\Theta}$ GMM parameters

1: **for** $t = 1, 2, \ldots$ **do**
2: Using $\mathcal{D}_t$, apply EM to find parameters $\mathbf{\Theta}_c^t$ $\forall c$ for the "labeled" GMM
3: Given unlabeled data, $\mathcal{B}_t$, which is sampled from $\mathcal{D}_t$ without the labeled information, apply EM to $\mathcal{B}_t$ to find parameters $\mathbf{\Theta}_{\mathcal{B}}^t$ for the "unlabeled" GMM
4: Classify $x_q \in \mathcal{B}_t$ using (3)
5: **end for**

Fig. 2. EM-based approach for learning a changing data stream (TRANSE-EM)

expressed as $K = \sum_c K_c$. In Line 4 of the pseudo code, a GMM, with parameters $\mathbf{\Theta}_{\mathcal{B}}^t$, is generated from the unlabeled data $\mathcal{B}_t$, when that data become available. This GMM with parameters $\mathbf{\Theta}_{\mathcal{B}}^t$ is formed with K components. Since $\mathcal{B}_t$ do not have labels, we cannot – yet – associate class labels to each component in the GMM with parameters $\mathbf{\Theta}_{\mathcal{B}}^t$. Therefore, we need to find a mechanism to relate the components in $\mathbf{\Theta}_c^t$ to $\mathbf{\Theta}_{\mathcal{B}}^t$.

Let us define an estimate of the posterior probability on $\mathcal{B}_t$ as $g_c(x)$, which is calculated in (3). As stated earlier, labels cannot be directly associated with each component in $\mathbf{\Theta}_{\mathcal{B}}^t$. However, using (3) we can assign a weight to each component in $\mathbf{\Theta}_{\mathcal{B}}^t$ for class c. This weight represents the amount of responsibility that class c accepts for representing component k in $\mathbf{\Theta}_{\mathcal{B}}^t$. This measure of responsibility is estimated as the posterior, $P(\omega_c|\mu_k)$, given the GMM generated from the labeled training data. The quantity $P(\omega_c|\mu_k)$ is easily computable with the GMMs with parameters $\mathbf{\Theta}_c^t$. The priors $P(\omega_c)$ is the prior probability that can be computed directly from the data. Given a limited amount of drift present between the source and target data, this estimate should reasonably indicate the class that is the most responsible for generating such data with μ_k.

$$g_c(x) = \frac{P(\omega_c) \sum_{k=1}^{K} P(\omega_c|\mu_k) \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{c'=1}^{C} P(\omega_{c'}) \sum_{k=1}^{K} P(\omega_{c'}|\mu_k) \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)} \tag{3}$$

where π_k are the GMM mixing coefficients, μ_k/Σ_k is the kth mean/covariance in $\mathbf{\Theta}_{\mathcal{B}}^t$ and $\mathcal{N}(x|\mu, \Sigma)$ is a Gaussian density function with mean μ and covariance Σ. The $P(\omega_c|\mu_k)$ can be computed using (4) with the parameters GMM parameters in $\mathbf{\Theta}_c^t$.

$$P(\omega_c|x) = \frac{P(\omega_c) \sum_{k=1}^{K_c} \pi_k^c \mathcal{N}(x|\mu_k^c, \Sigma_k^c)}{\sum_{c'=1}^{C} P(\omega_{c'}) \sum_{k=1}^{K_{c'}} \pi_k^{c'} \mathcal{N}(x|\mu_k^{c'}, \Sigma_k^{c'})} \tag{4}$$

where $\mathbf{\Theta}_c^t$ determines the GMM mixing coefficients π_k^c, the

covariance Σ_k^c, and the mean μ_k^c for the kth component of the cth class. If the drift is evolutionary in nature, then $P(\omega_c|\mu_k)$ provides a good inference of μ_k's class association.

The expected pseudo error for a hypothesis, h_l, $l = 1, \ldots, t$, is computed using (5) or (6), which is referred to as mean square error (mse) and absolute mean error (ame), respectively (see Line 5). The pseudo error is dependent on the probabilistic estimate of $g_c(x)$. In this work, we empirically test these two pseudo error measures, which are used to derive the classifier voting weights.

$$\epsilon_{l,t} = \frac{1}{n \cdot C} \sum_{c=1}^{C} \sum_{q=1}^{n} \left(g_c(x_q) - h_{l,c}(x_q)\right)^2 \tag{5}$$

$$\epsilon_{l,t} = \frac{1}{n \cdot C} \sum_{c=1}^{C} \sum_{q=1}^{n} |g_c(x_q) - h_{l,c}(x_q)| \tag{6}$$

where $\epsilon_{l,t}$ is the pseudo error of hypothesis l at time t, $h_{l,c}(x_i)$ is the posterior estimate for class c from hypothesis l on x_i and $g_c(x_i)$ is the GMM posterior for class c on x_i.

The classifier voting weights are determined in Line 6 using pseudo error described above. The form of voting weights is a commonly used form of a non-uniform weighting scheme inspired by Adaboost [23], and the final ensemble hypothesis, $H(x)$, is computed using a weighted majority vote as in Line 7.

A couple points are worth noting: first, h_l should preferably provide a continuous probabilistic interpretation of class supports; however, TRANSE can still run even if h_l only provides binary decisions. Second, it is reasonable to ask whether $H(x)$ should be used at all if $g_c(x)$ has strong influence on $H(x)$. This question – whether the ensemble members should be retained at all – has a complex answer, as we discuss below and again in the results section. The short answer is that decision depends on how accurate we believe our estimate of the posterior on $\mathcal{B}_t$ is. If this estimate is very accurate, then the ensemble members are not necessary, and we can directly use $g_c(x)$. A generalization of the TRANSE algorithm that follows this approach is shown in Fig. 2. This algorithm, which follows an EM-like mechanism - is referred to as TRANSE-EM.

We anticipate that the posterior estimate is likely to be accurate if indeed the GMM parameters (and the number of mixture components) have been accurately predicted, which is likely to happen for data with simple and well-behaving distributions. However, for complex distributions whose accurate estimate using the GMM may be difficult, the knowledge learned by the ensemble members – aided by the limited accuracy of the GMMs provided in the form of classier weights – can help build a robust knowledge base and continuously reinforce and update previously learned knowledge. Furthermore, note that the selection of GMM parameters (such as the user-defined number of mixture components) can also drastically change the effectiveness of the GMM-only approach as shown in Fig. 2. It is therefore important to evaluate challenging problems from real-world data to observe the impact and sensitivity of GMM parameters on the ability of the algorithm to track the changing environment. We provide preliminary results of such an analysis later in the results section of this paper.

B. Relationship Between Ensemble Error and the Probabilistic Model

In this section we present the error bound of ensemble hypothesis, and explore whether classifier weighting should be uniform or weighted. To guide us, we choose a natural measure of error between the classifier hypothesis, $h(x)$, and an ideal labeling function, $f(x)$, as the absolute difference given by (7), where $\mathbb{E}_{x\sim\mathcal{P}}[\cdot]$ is the expectation computed over some distribution $\mathcal{P}$ (note that $\mathcal{P}$ not a dataset).

$$\epsilon(h, f) = \mathbb{E}_{x\sim\mathcal{P}}[\mid h(x) - f(x) \mid] \tag{7}$$

This disagreement function can be computed between any arbitrary hypothesis h and labeling function f on a distribution $\mathcal{P}$. For the purpose of this paper, we are interested in $\epsilon_T(h_l, f_T)$, or $\epsilon_T(h_l)$ in shorthand notation, which is the disagreement between the hypothesis generated at time stamp l and the true labeling function on the target distribution, f_T. More importantly, we are interested in solving for the ensemble error bound on the target distribution, $\epsilon_T(H)$.

Theorem 1 (Classifier error bound): The error of a hypothesis, h_l, generated at time stamp l from $\mathcal{P}_l$, on a target $\mathcal{P}_T$ is bounded above by (8).

$$\begin{aligned}\epsilon_T(h_l) \leq \epsilon_l(h_l) + d_1(\mathcal{P}_l, \mathcal{P}_T) \\ + \min\{\epsilon_T(f_T, f_l), \epsilon_l(f_T, f_l)\}\end{aligned} \tag{8}$$

where $d_1(\cdot, \cdot)$ is the variational divergence. The proof of this theorem is given by Ben-David et al. [24].

Theorem 2 (Ensemble error bound): The error of the ensemble hypothesis, H, on a target $\mathcal{P}_T$ is bounded above by (9).

$$\epsilon_T(H) \leq \epsilon_T(H, g) + \epsilon_T(g) \tag{9}$$

Proof: The proof of Theorem 2 is rather straight forward and is proven by adding/subtracting $\epsilon_T(H, g)$, which is the error between the ensemble hypothesis and the GMM estimate of the true labeling function, as shown in below:

$$\begin{aligned}\epsilon_T(H) &= \epsilon_T(H, f_T) + \epsilon_T(H, g) - \epsilon_T(H, g) \\ &\leq \epsilon_T(H, g) + \mid \epsilon_T(H, f_T) - \epsilon_T(H, g) \mid \\ &\leq \epsilon_T(H, g) + \mathbb{E}_{x\sim\mathcal{P}_T}[\mid g(x) - f_T(x) \mid] \\ &\leq \epsilon_T(H, g) + \epsilon_T(f_T, g)\end{aligned}$$

■

The second term in the ensemble error bound, $\epsilon_T(f_T, g)$, is intuitive to understand as it is simply the disagreement between the probabilistic (GMM) model $g(x)$ and the true labeling function $f_T(x)$. However, the relationship of $g(x)$ and $H(x)$ has not been mathematically analyzed. Therefore, we should determine how $g(x)$ and $H(x)$ affect each other. The $\epsilon_T(H, g)$

term can be simplified by exploiting the fact that h_l and g can be written in terms of biased versions of f_T.

$$\begin{aligned}\epsilon_T(H,g) &= \mathbb{E}_{\mathcal{P}_T}[\mid g(x) - H(x)] \\ &= \mathbb{E}_{\mathcal{P}_T}\left[\left|g(x) - \sum_{l=1}^{t}\alpha_l h_l(x)\right|\right] \\ &= \mathbb{E}_{\mathcal{P}_T}\left[\left|f_T(x) + \beta(x) - \sum_{l=1}^{t}(w_l+\gamma_l)(f_T(x)+\tau_l(x))\right|\right]\end{aligned}$$

where $\tau_l(x)$ is h_l's bias (error) on x, $\beta(x)$ is $g(x)$'s bias on x, and γ_l is the bias of the calculated classifier weight (α_l) from the optimal voting weight w_l.

Let us assume for the moment $g \mapsto f_T \ \forall x$, that is, the GMM estimate approaches the true target function. If this is the case, then we have:

$$\lim_{g\to f_T}\epsilon_T(H,g) = \mathbb{E}_{\mathcal{P}_T}\left[\left|\sum_{l=1}^{t}\left\{\frac{1}{t}f_T(x) - w_l\underbrace{(f_T(x)+\tau_l(x))}_{h_l(x)}\right\}\right|\right]$$

since $\beta(x) \mapsto 0$ and $\gamma_l \mapsto 0$. The γ_l terms should be expected to decrease as g improves because the error of g affects the bias in the classifier weights. We can see that $f_T(x) - \sum_{l=1}^{t} w_l h_l(x) = 0$ is an obvious solution to the problem. Using the expectation over $\mathcal{P}_T$, our goal is to minimize $\mathbb{E}_{x\sim\mathcal{P}_T}[\mid f_T(x) - \sum_l w_l h_l(x) \mid]$. This analysis suggests that if we can infer upon the bias of a classifier, i.e., relate the classifier weights to the error of the classifier (as done in step 5 of the algorithm), weighted majority voting goes along with our intuition that poorly performing classifier should have their weights reduced. However, if this task is not possible, then using a uniform vote as suggested in [1] is the ideal choice. The experiments sections presents several algorithms that can infer upon bias and one that does not, thus indicating that a weighted majority vote may be better in those cases.

IV. Experiments

The proposed approach, TRANSE along with three variants, are compared against Learn^{++}.NSE (referred to as NSE in result tables) [25], SEA (streaming ensemble algorithm) [20], and WEA (weight estimation algorithm) [22]. The TRANSE approaches are refereed to as TRANSE (*ame*), (*mse*) and TRANSE-EM, where TRANSE (*ame*) uses (6) to compute expected risk, TRANSE (*mse*) uses (5) to compute expected risk and TRANSE-EM uses $g_c(x)$ as the hypothesis (i.e., ensemble classifiers are not used as presented in Fig. 2). WEA uses GMMs to generate synthetic data from the estimated target distribution to infer the expected error on the target. WEA uses unlabeled data to update the classifier ensemble in nonstationary environments. Learn^{++}.NSE generates a classifier when new data are presented. Classifiers in the ensemble are combined through a weighting scheme that accounts for a classifier's error in recent time [4], [25]. SEA is a benchmark ensemble algorithm that uses unweighted classifier voting [20]. The base classifier selected for all ensemble systems was the C4.5 decision tree. In each experiment we inject bias between the training and testing datasets as described in the next section. Several synthetic and real-world datasets have been selected to evaluate the algorithms.

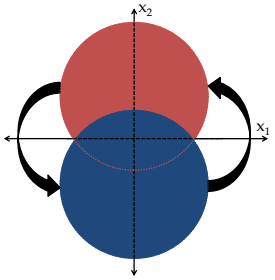

Fig. 3. The rotating Gaussian problem consists of two Gaussian probability density functions drift in a circular pattern causing environments to be encountered multiple times in a single experiment (i.e., a class expectation is controlled by $\mu^t = [\cos\theta_t, \sin\theta_t]^{\mathrm{T}}$, hence environments repeat 2π).

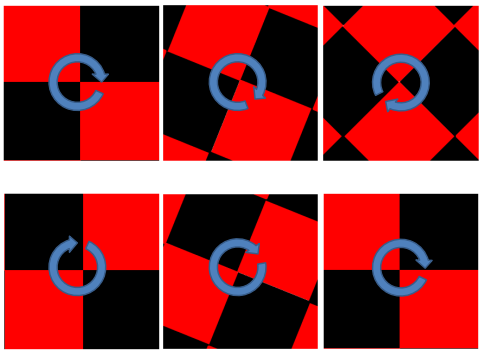

Fig. 4. Rotating checkerboard drifting scenario consists of a checkerboard pattern rotating between 0 and 2π in 300 evenly spaced intervals. Reoccurring environments are encountered at every integer multiple of π. Hence the data from 0 to π is identical to the data sample from π to 2π.

A. Datasets Used

We used two synthetic (rotating Gaussian and rotating checkerboard) and two real-world (Electricity Pricing and Weather) datasets to evaluate the proposed approach and compare its different versions to state-of-the-art approaches. The rotating Gaussian dataset, referred to as GaussCir, is comprised of two Gaussian components, each representing a different class, rotating around one another (Fig. 3). The class means are determined by using the parametric equations $\mu_1^t = [\cos\theta_t, \sin\theta_t]^{\mathrm{T}}$, $\mu_2^t = -\mu_1^t$ (where the subscripts 1 & 2 refer to classes), $\theta_t = 2\pi mt/N$, with fixed class covariance matrices given by $\Sigma_1 = \Sigma_2 = 0.5 * \mathbf{I}$, where m is the number of cycles, t is an integer valued time stamp ($t = 0, 1, \ldots, N-1$) and $\mathbf{I}$ is a 2×2 identity matrix. The experiment is run over 2 complete rotations with 1000 train/test instances in each batch, and with varying levels of bias between the training

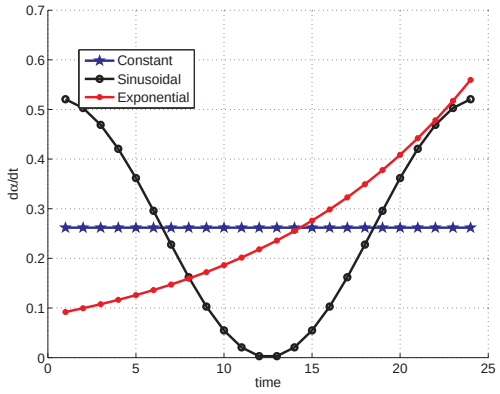

Fig. 5. The drift rate ($d\alpha/dt$) of the rotating checker boards varied according to the experiment in [4]. Predicting on the dataset becomes more difficult as the rate increases.

and testing datasets. Bias was injected by sampling the testing dataset from a future time stamp. In this paper, we refer to increasing number of time steps between training and test data as *increasing* or *injecting* bias.

The rotating checkerboard problem (RC) is a challenging generalization of the classical non-linear XOR problem[1]. The XOR problem is the special case when the angle of rotation equals $\alpha = 0$, $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$ and 2π. The experiment is controlled by two parameters, namely the relative side length of each square (representing one of the classes) with respect to the total length of the board (fixed at 0.5 in our experiments), and the angle of rotation, α. By varying the rotation angle in smaller steps between 0 and 2π, a significantly more challenging set of decision boundaries can be created. This experiment is particularly effective for observing an algorithm's ability to learn data in recurring environments. Fig. 4 depicts six snapshots of the board, corresponding to half of the rotating ($\alpha \in [0, \pi]$) checkerboard experiment. During the second half ($\alpha \in [\pi, 2\pi]$), the environment identically repeats the first half, which in turn simulates a recurring environment. We also experiment with three different drift rates in this work, namely: constant (RCC), exponential (RCE), and sinusoidal (RCS), as shown in Fig. 5.

The real-world Electricity Pricing dataset (elec2) was first presented in [26] and has been used as a benchmark for concept drift problems. This dataset is a sequence of information related to time and demand fluctuations in the price of electricity in New South Wales, Australia. We use the day, period, NSW (New South Wales) electricity demand, VIC (Victoria) electricity demand and the scheduled electricity transfer as the features. The task is to predict whether the NSW price will be higher or lower than VIC's in a 24 hour period.

Finally, the Weather Dataset is a subset of the NOAA dataset that we first processed and released as a concept drift dataset [27]. The dataset is obtained from Offutt Air Force Base in Bellevue, Nebraska. Daily measurements were taken for a variety of features such as temperature, pressure, visibility, and wind speed. We chose eight features and set the classification task to predict whether rain precipitation was observed on each day. The training size was set to 120 instances (days), approximately one season, and data from the next season served as the test data.

[1]The rotating checkerboard dataset among other resources and datasets for nonstationary environments can be downloaded from http://users.rowan.edu/∼polikar/RESEARCH/NSE/

TABLE I
RAW CLASSIFICATION ACCURACY AND RANKS FOR GAUSSIAN CIRCLE (GC) & REAL-WORLD DATASETS.

bias level	**NSE**	**SEA**	*ame*	*mse*	TRANSE-EM	WEA
GC 1	0.90 (3.5)	0.57 (6)	0.90 (3.5)	0.90 (3.5)	0.91 (1)	0.90 (3.5)
GC 2	0.88 (5)	0.51 (6)	0.89 (3)	0.89 (3)	0.91 (1)	0.89 (3)
GC 3	0.86 (5)	0.46 (6)	0.89 (2.5)	0.88 (4)	0.91 (1)	0.89 (2.5)
GC 5	0.80 (5)	0.35 (6)	0.87 (3)	0.87 (3)	0.91 (1)	0.87 (3)
GC 7	0.72 (5)	0.26 (6)	0.85 (3)	0.84 (4)	0.91 (1)	0.86 (2)
GC 10	0.54 (5)	0.16 (6)	0.75 (2.5)	0.67 (4)	0.84 (1)	0.75 (2.5)
GC 11	0.48 (5)	0.14 (6)	0.69 (2)	0.56 (4)	0.78 (1)	0.68 (3)
GC 13	0.35 (2)	0.11 (6)	0.28 (4)	0.23 (5)	0.37 (1)	0.31 (3)
GC 15	0.25 (1)	0.10 (6)	0.14 (4)	0.13 (5)	0.16 (2.5)	0.16 (2.5)
average	4.06	6.00	3.05	3.94	1.17	2.78
weather	0.76 (1)	0.75 (2)	0.71 (5)	0.73 (4)	0.64 (6)	0.74 (3)
electricity	0.70 (1)	0.67 (5)	0.68 (3)	0.68 (3)	0.64 (6)	0.68 (3)
average	1.0	3.5	4.0	3.5	6.00	3.00

TABLE II
RAW CLASSIFICATION ACCURACY AND RANKS FOR CHECKERBOARD DATASETS.

bias level	**NSE**	**SEA**	*ame*	*mse*	TRANSE-EM	WEA
RCS 1	0.73 (2)	0.59 (6)	0.71 (3.5)	0.68 (5)	0.78 (1)	0.71 (3.5)
RCS 5	0.63 (2)	0.55 (6)	0.61 (3.5)	0.59 (5)	0.67 (1)	0.61 (3.5)
RCS 10	0.58 (2)	0.52 (6)	0.56 (3)	0.54 (5)	0.62 (1)	0.55 (4)
RCS 15	0.59 (2)	0.49 (6)	0.57 (3.5)	0.55 (5)	0.61 (1)	0.57 (3.5)
RCS 25	0.50 (3.5)	0.49 (6)	0.50 (3.5)	0.50 (3.5)	0.52 (1)	0.50 (3.5)
RCS 50	0.83 (2)	0.64 (6)	0.78 (4)	0.76 (5)	0.85 (1)	0.79 (3)
RCC 1	0.73 (2)	0.57 (6)	0.70 (3.5)	0.68 (5)	0.77 (1)	0.70 (3.5)
RCC 5	0.61 (2)	0.49 (6)	0.59 (3.5)	0.57 (5)	0.65 (1)	0.59 (3.5)
RCC 10	0.51 (2)	0.45 (6)	0.50 (3.5)	0.48 (5)	0.55 (1)	0.50 (3.5)
RCC 15	0.41 (3)	0.46 (1)	0.39 (5)	0.38 (6)	0.45 (2)	0.40 (4)
RCC 25	0.56 (1)	0.41 (6)	0.55 (3)	0.52 (5)	0.55 (3)	0.55 (3)
RCC 50	0.81 (2)	0.66 (6)	0.76 (4)	0.72 (5)	0.85 (1)	0.77 (3)
RCE 1	0.71 (2)	0.58 (6)	0.69 (3.5)	0.65 (5)	0.77 (1)	0.69 (3.5)
RCE 5	0.59 (2)	0.52 (6)	0.58 (3.5)	0.56 (5)	0.66 (1)	0.58 (3.5)
RCE 10	0.53 (2)	0.49 (5)	0.50 (3.5)	0.48 (6)	0.58 (1)	0.50 (3.5)
RCE 15	0.48 (2)	0.47 (4)	0.47 (4)	0.46 (6)	0.52 (1)	0.47 (4)
RCE 25	0.43 (4)	0.44 (1)	0.43 (4)	0.43 (4)	0.43 (4)	0.43 (4)
RCE 50	0.82 (2)	0.65 (6)	0.81 (3)	0.77 (5)	0.86 (1)	0.79 (4)
average	2.19	5.28	3.61	5.03	1.33	3.58

B. Experimental Results

The preliminary results for TRANSE and their comparisons to other algorithms are presented in Table I and II for all datasets. We have grouped the results into two tables representing the Gaussian datasets (with bias) and the real-world datasets (as shown in Table II), and the rotating checkerboard variations (as shown in Table II) because of the large difference in database properties. For example, the Gaussian datasets are easily modeled with GMMs, whereas the rotating checkerboard is significantly more difficult to model because it is a uniform distribution with a fairly complex decision boundary. The (*ame*) and (*mse*) columns of the tables are TRANSE with the absolute mean error and mean square error weighting schemes applied to the ensemble, respectively. The TRANSE-EM column indicates the generalized version of TRANSE using $g_c(x)$ as the classifier as shown in Fig. 2. Fig. 6 shows that the accuracy of the proposed approaches remains near constant even though the bias between the training and testing data sets increases. Fig. 7 and 8 present a graphical

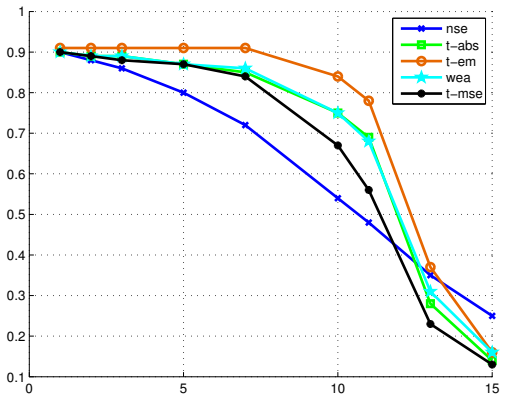

Fig. 6. Effect of increasing the bias between training and testing sets on a classifier's accuracy for the GaussCir dataset. As expected, all classifier's accuracy significantly drop as the bias between the training and testing datasets becomes extremely large; however algorithms that use unlabeled data are extremely effective when the bias is limited in nature.

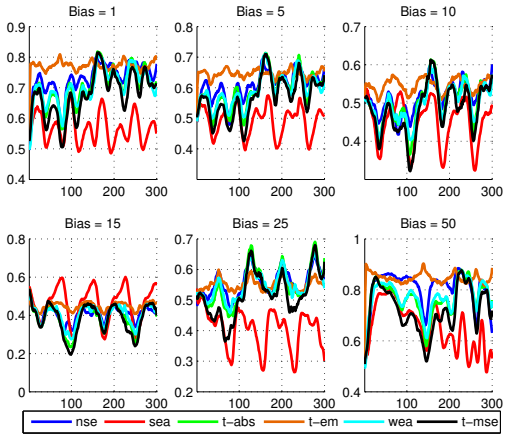

Fig. 7. Raw classification accuracy on the rotating checkerboard datasets with a constant drift rate over time.

representation of a classifier's raw classification accuracy over the duration of an experiment for the rotating checkerboard with constant drift and NOAA weather dataset, respectively. The first observation to note from Tables I & II and the figures is that weighted classifier voting schemes offer improved accuracy over uniform weighting. Furthermore, the approaches presented in this work and WEA, all of which use unlabeled data to tune ensemble parameters with unlabeled data, achieve higher accuracies than approaches (i.e. Learn^{++}.NSE and SEA) that simply predict on unlabeled data without any adjustments made for the unlabeled data.

Second, TRANSE-EM, TRANSE (*ame*), and TRANSE (*mse*) outperform ensembles that do not use unlabeled data (e.g., Learn^{++}.NSE and SEA) on the datasets where the bias is large, as in the Gaussian dataset with a 7 time stamp bias. This effect is also observed with WEA algorithm, which also uses unlabeled data. While Learn^{++}.NSE works well on a large array of problems; it performs behind TRANSE-EM,

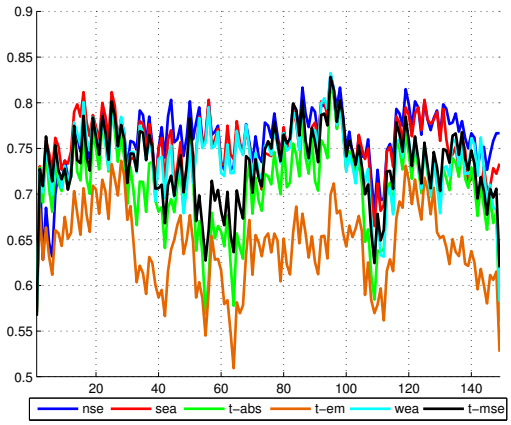

Fig. 8. Algorithm accuracy comparison on the weather dataset. The overall accuracy for each algorithm can be found in Table I. The TRANSE-EM approach, while highly effective on synthetic datasets performs poorly on a dataset where the cardinality of the data is low and $p(x)$ is not easily modelled.

but surprisingly performs better than TRANSE with (*ame*) or (*mse*) weighting in the overall average ranks. We believe Learn^{++}.NSE is outperforming the proposed approaches on the rotating checkerboard data because: a) Learn^{++}.NSE has been shown to be quite effective at recovering knowledge from the reoccuring environments and b) the GMM based approaches make a loose assumption that the decision boundaries formed by the labeling function $f_t(x)$ for any arbitrary time stamp are in regions of low density. Since the checkerboard data are distributed uniformly, this assumption is not met. Third, we see from Table I that TRANSE-EM works quite well when the datasets experience a large bias between the training and testing sources. We should note that the results for TRANSE-EM can vary greatly depending on the parameters selected (e.g., the number of components used in the GMM). Generating a good GMM can become increasingly difficult as the complexity of the true distribution increases. This is most dramatically illustrated with the results of the real world datasets (elec2 and weather), where the best TRANSE-EM was significantly outperformed by other versions of TRANSE and other approaches (refer to Table I).

V. Discussion & Conclusion

In this proof-of-concept study we investigated the use of unlabeled data to essentially "tune" a classifier to be a better predictor on data, which can be particularly effective in a transductive learning setting for nonstationary environments. The approaches presented in this paper and in [22] are robust to changes in the source used for testing a classifier. In this work, we selected a transductive methodology for classification rather than semi-supervised learning because developing a generalized mapping function for (distant) future data may be unfeasible for future unlabeled data as the source generating the data is drifting over time. Therefore, our approaches have focused on adapting the classifier to best classify the unlabeled data that are currently available. We presented a general

framework for using labeled and unlabeled data to aid in configuring and fine tuning an ensemble classifier's decision. GMMs are generated from labeled and target (unlabeled) data, which allows for the transfer of class information onto the unlabeled distribution. Once the class information of the target distribution has been inferred, classifier voting weights are updated using this information and the ensemble decision is made on the target. Results indicate that the GMMs are all that are needed on simple problems, whereas complex problems (e.g., where the probability distribution is not easily modeled) are aided by retaining ensemble member classifiers. Our results have also shown that the simple tranductive algorithm, which does not use the classifier ensemble, can be effective for moderately biased datasets provided that there is sufficient data to form a reliable GMM.

While the results for TRANSE-EM are appealing, we emphasize that in a practical setting (e.g., limited data and a complex distribution that is not easily modeled), TRANSE-EM becomes infeasible. In such complex cases, the ensemble, such as TRANSE with mse or ame, approach combined with weighted majority voting, where the weights are based on the GMM based probabilistic model, clearly outperforms the non-ensemble approaches. For simpler cases, such as the rotating Gaussians, a simpler version of the TRANSE approach where we simply use the $g_c(x)$ instead of the ensemble is also effective, as $g_c(x)$ is a good approximator of the class posteriors in such cases. Using the $g_c(x)$ alone is not recommended for complex real world cases, however, because of the difficulty in accurately estimating the GMM parameters. In such cases, the imperfect posteriors estimated by the GMM are not sufficient to track the changing environment, however can be extremely useful in guiding the ensemble to properly weight its classifiers to effectively track the changing environment. Approaches that keep an ensemble of discriminative classifiers such as TRANSE with either ame or mse options (and even Learn^{++}.NSE), can significantly improve over TRANSE-EM. Furthermore, many application driven problems that involve incremental learning require that model adaptation be implemented, and from our experiments, unlabeled data should be used to refine a classifier before field data are classified. If a small amount of labeled data are available from the field data, then more advanced transductive and/or domain adaptation approaches should be investigated. We leave this final note as expansion for future work.

REFERENCES

[1] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *SIAM International Conference on Data Mining*, 2007, pp. 203–208.

[2] K. Yamauchi, "Incremental learning and model selection under virtual concept drifting environments," in *Int'l Joint Conf. on Neural Netw.*, 2010, pp. 1–8.

[3] A. Kuh, T. Petsche, and R. L. Rivest, "Learning time-varying concepts," in *NIPS*, 1991.

[4] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. on Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, 2011.

[5] M. Baena-Garcia, J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavaldua, and R. Morales-Bueno, "Early drift detection method," in *Int'l Wksp. on Knowl. Disco. from Data Streams*, 2006.

[6] G. Ditzler and R. Polikar, "Hellinger distance based drift detection for nonstationary environments," in *IEEE Symp. on Comp. Intell. in Dyn. and Uncert. Envir.*, 2011, pp. 41–48.

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

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

[9] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers–part I: Detecting nonstationary changes," *IEEE Trans. on Neural Netw.*, vol. 19, no. 7, pp. 1145–1153, 2008.

[10] ——, "Just-in-time adaptive classifiers–part II: Designing the classifier," *IEEE Trans. on Neural Netw.*, vol. 19, no. 12, pp. 2053–2064, 2008.

[11] C. Alippi, G. Boracchi, and M. Roveri, "Just in time classifiers: managing the slow drift case," in *Int'l Joint Conf. on Neural Netw.*, 2009, pp. 114–120.

[12] C. Alippi and M. Roveri, "An effective just-in-time adaptive classifier for gradual concept drifts," in *Int'l Joint Conf. on Neural Netw.*, 2011, pp. 1675–1682.

[13] I. Žliobaitė, "Combining similarity in time and space for training set formation under concept drift," *Intelligent Data Analysis*, vol. 15, no. 4, p. to appear, 2010.

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

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

[16] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. on Knowl. and Data Engr.*, vol. 23, no. 6, pp. 859–874, 2011.

[17] A. Bifet, G. Holmes, B, Pfahringer, R. Kirkby, and R. Gavalda, "New ensemble methods for evolving data streams," in *Knowl. and Data Disc.*, 2009.

[18] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. on Knowl. and Data Engr.*, vol. 22, no. 5, pp. 731–742, 2010.

[19] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. on Knowl. and Data Engr.*, 2011.

[20] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large scale classification," in *ACM SIGKDD Int'l Conf. on Knowl. Disc. & Data Mining*, 2001, pp. 377–382.

[21] P. Zhang, X. Zhu, J. Tan, and L. Guo, "Classifier and cluster ensembles for mining concept drifting data streams," in *Int'l Conf. on Data Mining*, 2010, pp. 1175–1180.

[22] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," in *Int'l Joint Conf. on Neural Netw.*, 2011, pp. 2471–2478.

[23] Y. Freund and R. Shapire, "A short introduction to boosting," *Journal of Japanese Soc. for Artif. Intell.*, vol. 14, no. 5, pp. 771–780, 1999.

[24] S. Ben-David, J. Blitzer, K, Crammer, A. Kulesza, P. Pereira, and J. Wortman Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, pp. 151–175, 2010.

[25] M. Muhlbaier and R. Polikar, "Multiple classifiers based incremental learning algorithm for learning nonstationary environments," in *Int'l Conf. on Mach. Learn. and Cybern.*, 2007, pp. 3618–3623.

[26] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," The University of South Wales, Tech. Rep., 1999.

[27] United States Department of Commerce, "National oceanic and atmospheric administration," 2010. [Online]. Available: www.noaa.gov/