Hellinger Distance Based Drift Detection for Nonstationary Environments

Gregory Ditzler and Robi Polikar
Dept. of Electrical & Computer Engineering
Rowan University
Glassboro, NJ, USA
gregory.ditzer@gmail.com, polikar@rowan.edu

***Abstract*—Most machine learning algorithms, including many online learners, assume that the data distribution to be learned is fixed. There are many real-world problems where the distribution of the data changes as a function of time. Changes in nonstationary data distributions can significantly reduce the generalization ability of the learning algorithm on new or field data, if the algorithm is not equipped to track such changes. When the stationary data distribution assumption does not hold, the learner must take appropriate actions to ensure that the new/relevant information is learned. On the other hand, data distributions do not necessarily change continuously, necessitating the ability to monitor the distribution and detect when a significant change in distribution has occurred. In this work, we propose and analyze a feature based drift detection method using the Hellinger distance to detect gradual or abrupt changes in the distribution.**

Keywords-concept drift; nonstationary environments; drift detection

I. Introduction

Detecting change in data distributions is an important problem in data mining and machine learning algorithms due to increasing number of applications that are governed by such data. Any algorithm that does not make the necessary adjustments to changes in data distribution will necessarily fail to provide satisfactory generalization on future data, if such data do not come from the distribution on which the algorithm was originally trained. For example, consider an application that tracks a user's web browsing habits to determine which ads are most relevant for that user's interest. User interests are known to change – or *drift* – over time. In such cases, certain ads in which the customer used to express interest may no longer be relevant. Thus, an algorithm designed to determine the relevant ads must be able to monitor the customer's browsing habits and determine when there is change in the customer's interest. Applications that call for an effective change or drift detection algorithm can be expanded: analysis of electricity demands or pricing, financial or climate data are all examples of applications with nonstationary data distributions, where change or drift detection is needed so that the learner can take an appropriate action.

For the purposes of this paper, we define a sudden or abrupt change in the underlying data distribution that alters the decision boundaries as a *concept change*, whereas gradual changes in the data distribution as a *concept drift*. However, when the context does not require us to distinguish between the two, we use the term concept drift to encompass both scenarios, as it is usually the more difficult one to detect.

Learning from drifting environments is usually associated with a stream of incoming data, either one instance or one batch at a time. There are two types of approaches for drift detection in such streaming data: in *passive drift detection*, the learner assumes – every time new data become available – that some drift may have occurred, and updates the classifier according to the current data distribution, regardless whether drift actually did occur. In *active drift detection*, the algorithm continuously and explicitly monitors the data to detect if and when drift occurs. If – and only if – the drift is detected, the algorithm takes an appropriate action, such as updating the classifier with the most recent data or simply creating a new classifier to learn the current data. The drift detection method presented in this work is appropriate for an active drift detection framework.

This paper is organized as follows: Section II provides an overview of drift detection algorithms, followed by our motivation for using Hellinger distance as a metric, as well as a detailed explanation of the approach for the proposed algorithm in Section III. Section IV presents results obtained by the proposed approach on several real world & synthetic datasets. Finally, Section V provides concluding remarks and future work.

II. Background

Concept drift algorithms are usually associated with incremental learning of streaming data, where new datasets become available in batches or in an instance-by-instance basis, resulting in batch or online learning, respectively [1]. Given such data, the (active) drift detection itself can be parametric or non-parametric, depending on whether a specific underlying distribution is assumed. Many parametric algorithms use a CUSUM (cumulative sum) based mechanism, which is traditionally used for control charts in detecting nonstationary changes in process data [2-4]. A series of successful implementations of this approach are proposed by Alippi & Roveri, including CI-CUSUM [5;6], a *pdf* free extension of the traditional CUSUM. Recently, Alippi et al. also introduced the

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

intersection of confidence intervals (ICI) rule for drift detection [7]. Some drift detection approaches such as the Early Drift Detection Method (EDDM) [8] and similar approaches [9], do not make any assumptions on feature distribution, but rather monitor a classifiers' accuracy or some distance metric to detect drift. Cielslak & Chawla suggest Hellinger distance, not for detecting concept drift in an incremental learning setting, but rather to detect bias between training and test data distributions [10]. Using a non-parametric statistical test, the authors measure the significance between the probability estimates of the classifier on a validation set (carved out from training data) and the corresponding test dataset [10]. To do so, a baseline comparison is first made by calculating the Hellinger distance between the original training and test datasets. Bias is then injected into the testing set. A baseline Hellinger distance (i.e. when no bias is present between training/testing sets) and the distance after bias is injected into the dataset are observed.

We extend this approach to concept drift in an incremental learning setting, where new datasets are presented in batches over time. We do not use the original dataset as the baseline distribution against which other data distributions are compared, nor do we inject any bias into future distributions, as done in [10], but rather we monitor the magnitude of the change in Hellinger distance between a new distribution and a baseline distribution, which is updated every time drift is detected.

III. Approach

We begin this section by introducing the motivation for using the Hellinger distance as a measure that can be applied to drift detection in an incremental learning setting. Next, we present the proposed Hellinger Distance Drift Detection Method (HDDDM). According to four criteria suggested by Kuncheva [3], HDDDM can be categorized as follows:

- Data chunks: batch based
- Information used: raw features (not based on classifier performance)
- Change detection mode: explicit
- Classifier-specific vs. classifier-free: classifier free

The algorithm uses a hypothesis testing-like statistical approach to determine if there is enough evidence to suggest that the data instances are being drawn from different distributions.

A. *Motivation*

The primary motivation for this detection method is to detect if drift is present in a sequence of batch data. While prior work used the Hellinger distance only as a measure between a single training and testing set, we extend the approach to sequential batch learning where the goal is to detect drift among different datasets. We select the Hellinger distance over other measures such as the Mahalanobis distance, because there are no assumptions made about the distribution of the data. Also, the Hellinger distance is a measure of distributional divergence which will allow HDDDM to measure the change between the distributions of data at two subsequent time stamps.

In contrast to EDDM and other similar approaches that rely on classifier error [8;9], the proposed Hellinger distance drift detection method (HDDDM) is a feature based drift detection method, using the Hellinger distance between current data distribution and a reference distribution that is updated as new data are received. The Hellinger distance is an example of f divergence measure, similar to the Kullback-Leibler (KL) divergence. However, unlike the KL-divergence the Hellinger divergence is a symmetric metric. Furthermore, unlike most other distance metrics, the Hellinger distance is a bounded distance measure: for two distributions with probability mass functions (or histograms representing these distributions) P and Q, the Hellinger distance is $\delta_H(P,Q) \in [0,\sqrt{2}]$. If $\delta_H(P,Q) = 0$, the two probability mass functions are completely overlapping and hence identical. If $\delta_H(P,Q) = \sqrt{2}$, the two probability mass functions are completely divergent (i.e. there is no overlap).

As an example, consider a two-class rotating mixture of Gaussians with class centers moving in a circular pattern (Fig. 1), with each distribution in the mixture corresponding to a different class label. The class means can be given by the parametric equations $\mu_1^{(t)} = [\cos\theta_t, \sin\theta_t]^T$, $\mu_2^{(t)} = -\mu_1^{(t)}$, $\theta_t = \frac{2\pi c}{N}t$, with fixed class covariance matrices given as $\Sigma_1 = \Sigma_2 = 0.5 * \mathbf{I}$, where c is the number of cycles, t is the (integer valued) time stamp that iterates from zero to $N-1$, and $\mathbf{I}$ is a 2x2 identity matrix. Fig. 2 shows the evolution of the Hellinger distance computed between the datasets ($\mathcal{D}_k$) generated with respect to θ_1 and θ_k where $k = 2,\dots,N-1$. The Hellinger distance is capable of displaying the relevance or the closeness of a new dataset ($\mathcal{D}_k$) to a baseline dataset ($\mathcal{D}_1$) as shown in Fig. 2. We plot the Hellinger distance for the divergence of the datasets for class ω_1, class ω_2, and the entire data separately. The Hellinger distance varies as θ begins to evolve. We observe that when θ_1 and θ_k are the same (or very similar) the Hellinger distance is small as observed at $t = \{0, 200, 400, 600\}$. This is when $\mu_1^{(t)} = \mu_1^{(1)}$ and $\mu_2^{(t)} = -\mu_1^{(t)} = -\mu_1^{(1)}$. We observe that the Hellinger distance computed for all data (entire dataset) repeats every 100 time stamps, twice as often compared to class specific distributions,

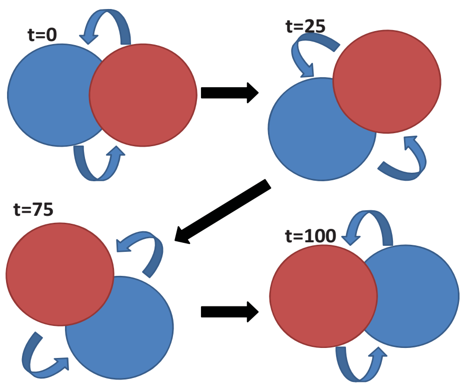

Fig. 1. Evolution of a binary classification task with two drifting Gaussian distributions.

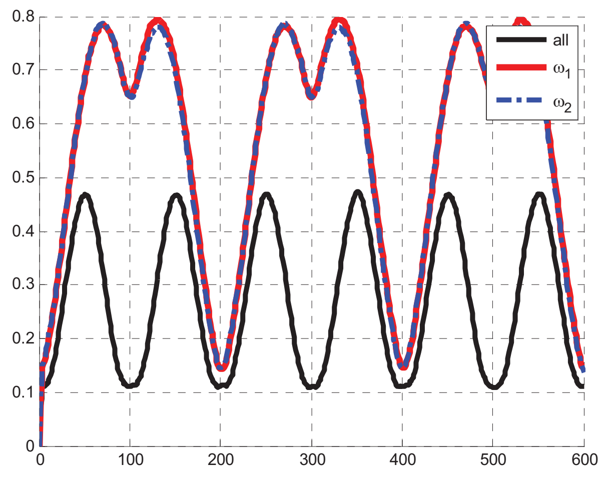

Fig. 2. Hellinger distance (vertical axis, plotted against time) computed on a rotating Gaussian centers problem. The Hellinger distance is computed between datasets $\mathcal{D}_1$ and $\mathcal{D}_t$ where $t = 2,3,\dots,600$ for ω_1, ω_2, and all classes. Recurring environments occur when the Hellinger distance is at a minimum.

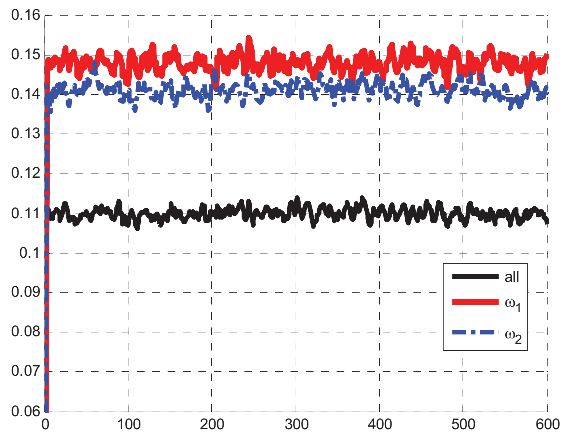

Fig. 3. Hellinger distance computed on a static Gaussian problem. The Hellinger distance is computed between $\mathcal{D}_1$ and $\mathcal{D}_t$ where $t = 2,3,\dots,600$ for ω_1, ω_2, and all classes.

which repeat every 200 time steps. This is because, as seen in Fig. 1, every 100 time steps, the data consists of the exact same instances at the exact same locations, but with their labels flipped, compared to the distribution at $t = 0$.

As the class centers evolve from $t = 0$ to $t = 200$, we observe changes in Hellinger distances that follow the expected trends based on Fig. 1, reaching the maximum value at $t = 75$ and $t = 125$, with a slight dip at $t = 100$, for the class-specific datasets. The Hellinger distance then reduces as the class distributions return to their initial state at $t = 200$. The entire scenario then repeats two more times.

If the distributions governing the data at different time stamps is static (i.e. $\theta =$ constant), then the Hellinger distance between the 1st batch and subsequent batches remains constant as shown in Fig. 3. We note that a static distribution does not mean zero Hellinger distance. In fact, the Hellinger distance will be non-zero due to differences in class means as well as the random nature of the data drawn for each sample. The Hellinger distance remains near constant; however, as the distribution itself does not change.

Having observed that the Hellinger distance does change between two distributions as these two distributions diverge from each other, and remain constant if they do not, we can use this information to determine when change is present in an incremental learning problem. The process of tracking the data distributions for drift is described below.

B. *Assumptions*

The proposed approach makes three assumptions: i) labeled training datasets are presented in batches to the drift detection algorithm, as the Hellinger distance is computed between two histograms (of distributions) of data. If only instance based streaming data is available, one may accumulate instances to form a batch to compute the histogram; ii) data distributions have finite support (range): $P(X \le x) = 0$ for $x \le T_1$ and $P(X \ge x) = 0$ for $x \ge T_2$, where $T_1 < T_2$ are finite real numbers. We fix the number of bins in the histogram required to compute the Hellinger distance at $\lfloor \sqrt{N} \rfloor$, where N is the number of instances at each time stamp presented to the drift detection algorithm. This can be manually adjusted if one has prior knowledge to justify otherwise. Under minimal or no prior knowledge, $\lfloor \sqrt{N} \rfloor$ works well. iii) Finally, in order to follow a true incremental learning setting, we assume that we do not have access to old data [11]. Each instance is only seen once by the algorithm.

C. *Drift Dection Algorithm*

The pseudo code of the proposed Hellinger distance drift detection method (HDDDM) is shown in Fig. 4. As mentioned above, we assume that the data arrive in batches, with dataset $\mathcal{D}_t$ becoming available at time stamp t. The algorithm initializes $\lambda = 1$ and $\mathcal{D}_\lambda = \mathcal{D}_1$ where λ indicates the last time stamp in which change was detected. $\mathcal{D}_1$ is established as the first baseline reference dataset to which we compare future datasets for possible drift. This baseline distribution, $\mathcal{D}_\lambda$, is updated incrementally as described below.

The algorithm begins by constructing histogram P from $\mathcal{D}_t$ and histogram Q from $\mathcal{D}_\lambda$ with $b = \lfloor \sqrt{N} \rfloor$ bins, where N is the cardinality of the current data batch presented to the algorithm. The Hellinger distance between the two histograms P and Q is then computed using Eq. (1). The (intermediate) Hellinger distance is calculated first for each feature, and the average of distances for all features is then considered as the final Hellinger distance.

$$\delta_H(t) = \frac{1}{d}\sum_{k=1}^{d} \sqrt{\sum_{i=1}^{b} \left(\sqrt{\frac{P_{i,k}}{\sum_{j=1}^{b} P_{j,k}}} - \sqrt{\frac{Q_{i,k}}{\sum_{j=1}^{b} Q_{j,k}}} \right)^2} \qquad (1)$$

where d is the dimensionality of the data, and $P_{i,k}$ ($Q_{i,k}$) is the frequency count in bin i of the histogram corresponding to the histogram P (Q) of feature k. We then compute $\epsilon(t)$, the difference in divergence between the most recent measure of the Hellinger distance, $\delta_H(t)$, and the Hellinger distance measured at the previous time stamp, $\delta_H(t-1)$. This differential between the current and prior distances is compared to a threshold, to determine whether the change is large enough to

```
Input: Training data, $\mathcal{D}_t = \left\{\boldsymbol{x}_i^{(t)} \in X^{(t)}; y_i \in \Omega\right\}$, presented in
        batches corresponding to a joint probability distribution
        $p_t(\boldsymbol{x}, \omega)$ where $t = 1,2,\ldots$
        Initialize: $\lambda = 1$, and $\mathcal{D}_\lambda = \mathcal{D}_1$
for $t = 2,3,\ldots$ do
  1. Generate a histogram, $P$, from $\mathcal{D}_t$ and a histogram, $Q$, from
     $\mathcal{D}_\lambda$. Each histogram has $b = \lfloor\sqrt{N}\rfloor$ bins, where $N$ is the
     cardinality of $\mathcal{D}_t$
  2. Calculate the Hellinger distance between $P$ and $Q$ using Eq.
     (1). Call this $\delta_H(t)$.
  3. Compute the difference in Hellinger distance
         $\epsilon(t) = \delta_H(t) - \delta_H(t-1)$
  4. Update the adaptive threshold
         $\hat{\epsilon} = \frac{1}{t-\lambda-1}\sum_{i=\lambda}^{t-1}|\epsilon(i)|$
         $\hat{\sigma} = \sqrt{\frac{\sum_{i=\lambda}^{t-1}(|\epsilon(i)| - \hat{\epsilon})^2}{t-\lambda-1}}$
     Compute $\beta(t)$ using the standard deviation in Equation (2) or
     the confidence interval method in Equation (3).
  5. Determine if drift is present
     if $|\epsilon(t)| > \beta(t)$ then
        $\lambda = t$
        Reset $\mathcal{D}_\lambda$ by setting $\mathcal{D}_\lambda = \mathcal{D}_t$
        Indicate change was detected
     else
        Update $\mathcal{D}_\lambda$ with $\mathcal{D}_t \rightarrow \mathcal{D}_\lambda = \{\mathcal{D}_\lambda, \mathcal{D}_t\}$
     end if
end for
```

Fig. 4. Algorithm pseudo code for the Hellinger distance drift detection method (HDDDM)

claim a drift. Rather than heuristically or arbitrarily selecting a threshold, we use an adaptive threshold that is automatically adjusted at each time stamp. To do this, we first compute $\hat{\epsilon}$, the mean $|\epsilon(i)|$, and $\hat{\sigma}$, the standard deviation of differences in divergence, where $i = \lambda, \lambda+1, \ldots, t-1$. Note that the current time stamp and all steps before λ (last time a change was detected) are not included in the mean difference calculation.

The actual threshold $\beta(t)$ at time step t is then computed based on the mean and standard deviations of the differences in divergence. We propose two alternatives to compute this threshold: based on the standard deviation and on a confidence level. The first is computed simply by

$$\beta(t) = \hat{\epsilon} + \gamma\hat{\sigma} \tag{2}$$

where γ is some positive/real constant, indicating how many standard deviations of change around the mean we accept as "different enough". Note that we use $\hat{\epsilon} + \gamma\hat{\sigma}$ and not $\hat{\epsilon} \pm \gamma\hat{\sigma}$, as we flag a drift when the magnitude of the change is *significantly greater* than the average of the change in recent time (since the last detected change) with the significance controlled by the γ term and the standard deviation of the divergence differences.

The second implementation of HDDDM uses the t-statistic and scales by the square root of $t-\lambda-1$. Again, an upper-tailed test is used, $\hat{\epsilon} + t_{\alpha/2,df}\frac{\hat{\sigma}}{\sqrt{t-\lambda-1}}$ and not the interval $\hat{\epsilon} \pm t_{\alpha/2,df}\frac{\hat{\sigma}}{\sqrt{t-\lambda-1}}$.

$$\beta(t) = \hat{\epsilon} + t_{\alpha/2,df}\frac{\hat{\sigma}}{\sqrt{t-\lambda-1}} \tag{3}$$

Once the adaptive threshold is computed, it is applied to observed current difference in divergence to determine if drift is present in the most recent data. If magnitude $|\epsilon(t)| > \beta(t)$, then we signal that change has been detected in the most recent batch of data. As soon as change is detected, the algorithm *resets* $\lambda = t$ and $\mathcal{D}_\lambda = \mathcal{D}_t$. Note that the resetting of the baseline distribution is essential as the old distribution is no longer an accurate reference to determine how much the data distribution has changed (and whether that change is significant) as indicated by the large difference in Hellinger distance between time stamps. If drift is not detected, then we *update*, rather than reset, the distribution $\mathcal{D}_\lambda$ with the new data $\mathcal{D}_t$. Thus, $\mathcal{D}_\lambda$ continues to be updated with the most recent data as long as drift is not detected. This histogram can be updated or reset using the following equation:

$$\begin{aligned} Q_{i,k} &\leftarrow Q_{i,k} + P_{i,k}, \quad \text{if drift is not detected} \\ Q_{i,k} &\leftarrow P_{i,k}, \quad \text{if drift is detected} \end{aligned}$$

Note that the normalization in Eq. (1) ensures that the correct density is obtained from these histograms.

The algorithm naturally lends itself to an incremental drift detection scenario since it does not need access to previously seen data. Rather, only the histogram of the current and reference distributions are needed. Therefore, the only memory required for this algorithm is the previous values of ϵ, $\delta_H(t-1)$ and the histogram of $\mathcal{D}_\lambda$ (Q).

D. Algorithm Performance Assessment

Determining the effectiveness of a drift detection algorithm can be best tested on carefully designed synthetic datasets using a variety of drift scenarios, such as abrupt or gradual drift. Since the drift is deliberately inserted into the dataset, the ability of the algorithm to detect a drift when one is known to exist (measured by sensitivity), as well as its ability to not signal one when there is in fact no drift (measured by specificity), can be easily determined for such datasets.

Detecting drift is only useful to the extent such information can be used to guide a learner track a nonstationary environment and learn the drift in the data. To do so, we use a naïve Bayes classifier, which can be easily updated as new data arrive to determine whether detection of drift can be used to improve the classifier's performance on a drifting classification problem. We use the following procedure to determine the effectiveness of the drift detection algorithm:

- Generate two naïve Bayes classifiers with the 1st database. Call them $\mathcal{N}_1$ and $\mathcal{N}_2$.

`FOR` $t = 2,3,\ldots$

- Begin processing new databases for the presence of concept drift using HDDDM.
  - $\mathcal{N}_1$ is our target classifier which is updated or reset based on HDDDM decision. If drift is detected using HDDDM, we reset $\mathcal{N}_1$ and train $\mathcal{N}_1$ only on the

```
        new data, otherwise incrementally update $\mathcal{N}_1$ with
        the new data.
    –   Regardless of whether or not drift is detected, $\mathcal{N}_2$
        is incrementally trained with the new data. $\mathcal{N}_2$ is
        therefore the control classifier that is *not* subject to
        HDDDM intervention.
    –   Compute error of $\mathcal{N}_1$ and $\mathcal{N}_2$ on new test data
ENDFOR
```

We expect $\mathcal{N}_1$ to outperform $\mathcal{N}_2$ on drifting environments.

The online implementation of the naïve Bayes classifier is straightforward as the features are assumed class conditionally independent. The calculation of $p(x_i|\omega_j)$ required to implement naïve Bayes classifier, can be further simplified by assuming the data for the ith feature is normally distributed (though, this is not necessary for the algorithm to work). Thus, the parameters of naïve Bayes are computed as:

$$p(x_i|\omega_j) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right) \tag{4}$$

$$P(\omega_j|\boldsymbol{x}_i) \propto P(\omega_j) \prod_{i=1}^{d} p(x_i|\omega_j) \tag{5}$$

where x_i is the ith feature, σ_{ij}^2 is the variance of the ith feature of the jth class, μ_{ij} is the mean of the ith feature of the jth class, and ω_j is the label of the jth class.

IV. Experiments

Several synthetic and real world datasets, described below, were selected to evaluate the HDDDM algorithm.

A. Description of Datasets

As summarized in Table I, seven datasets were used. The *rotating checkerboard* dataset is generated using the Matlab source found in [12], and used in two scenarios controlled by changing the rotation of the checkerboard: i) continuous rotation over 400 time stamps for gradual drift and ii) discrete rotations every 20 of 300 time steps (a total of 15 changes) to simulate abrupt changes.

TABLE I. Description of datasets and their drift properties used for the experimentation of the HDDDM algorithm.

Dataset	Instances	Source	Drift type
Checkerboard	800,000	Synthetically Generated	Synthetic
Elec	27,549[4]	Web source[1]	Natural
SEA [5%]	400,000	Synthetically Generated	Synthetic
RandGauss	812,500	Synthetically Generated	Synthetic
Magic	13,065	UCI[2]	Synthetic
NOAA	18,159	NOAA[3]	Natural
GaussCir	950,000	Synthetically Generated	Synthetic

1. http://www.liaad.up.pt/~jgama/ales/ales_5.html
2. UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/)
3. National Oceanic and Atmospheric Administration(www.noaa.gov)
4. All missing instances with missing features have been removed

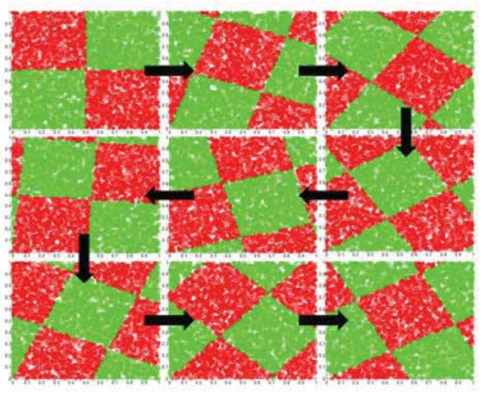
Fig. 5. Evolution of the rotating checkerboard dataset

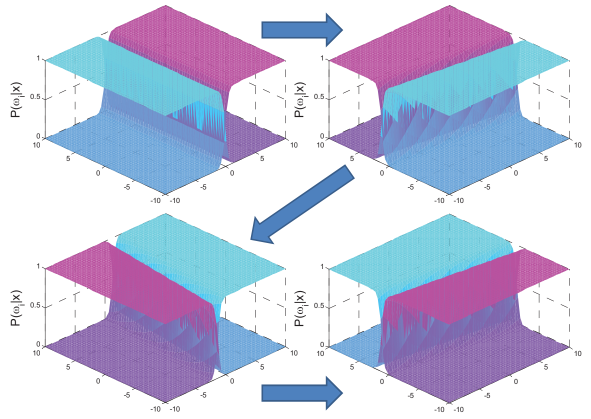

Fig. 6. Evolution of the circular Gaussian drift by observing the posterior probability. The drift is continuous from one batch to the next and not abruptly changing. There are 100 time stamps for each cycle.

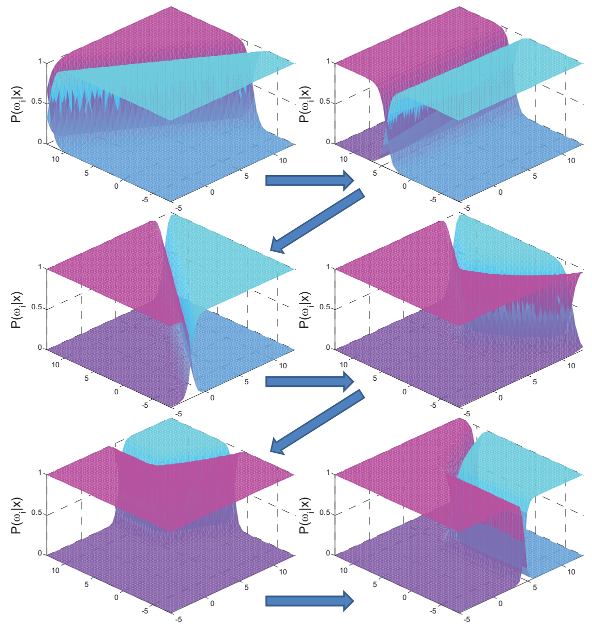

Fig. 7. Evolution of the RandGauss dataset by observing the posterior. The dataset begins with 2-modes (one for each class) and begins slowly drifting. The drift stops at the third evolution point and remains static for 25 time stamps followed by the introduction of a new mode for the magenta class. The dataset continues to evolve with slow change followed by abrupt changes.

Figure 5 illustrates a few snapshots of the checkerboard appearances over time. Electricity pricing (Elec) is a real-world dataset that contains natural drift within the database. SEA dataset comes from the original shifting hyper-plane problem presented by Street & Kim [13]. Magic dataset is from the UCI machine learning repository [14]. The original Magic dataset includes little, if any, drift. Therefore, this dataset has been modified by sorting a feature in ascending order and then generating incremental batches on the sorted data with a meaningful drift as an end-effect.

GaussCir and RandGauss are two synthetic datasets generated with a controlled drift scenario (whose decision boundaries are shown in Fig. 6 and 7, respectively). GaussCir is the example previously described in Section III. The NOAA dataset contains approximately 50 years of weather data obtained from a post at Offutt Air Force Base in Bellevue, Nebraska. Daily measurements were taken for a variety of features like temperature, pressure, visibility, wind speed, etc. The number of feature vectors was reduced to eight features and the classification task was to predict whether or not there was rain on a particular day.

B. Preliminary Results

The preliminary results are summarized in Fig. 8. Each of the plots presents the error of a naïve Bayes classifier with no update or intervention from HDDM, and with various values of γ (0.5, 1.0, 1.5, 2.0) for the standard deviation implementation of the HDDDM, as well as with α=0.1 for the confidence interval based implementation of HDDDM. The primary observation we make is that the classifier that is continuously updated, disregarding the concept change (red curve, corresponding to $\mathcal{N}_2$, i.e., no drift detection being used), has an overall error that is typically greater than the error of the classifiers that are reset based on the intervention of the HDDDM (other colors, corresponding to different implementations of $\mathcal{N}_1$). We would like to note that resetting an online classifier is

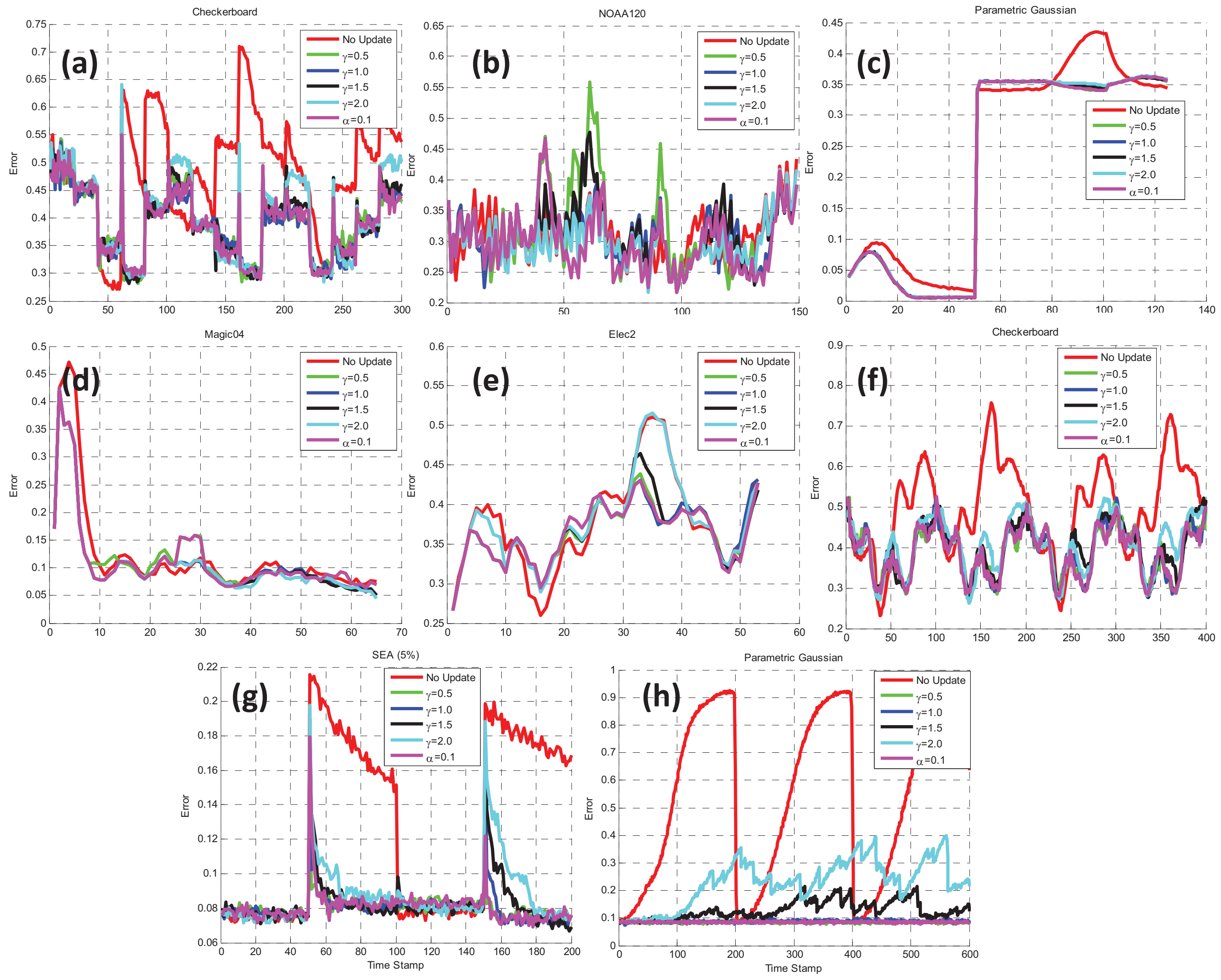

Fig. 8. Error evaluation of the online naïve Bayes classifiers (updated and dynamically reset) with a variation in the parameters of the Hellinger Distance Drift Detection Method (HDDDM). (a) Checker board with abrupt changes in rotation, (b) NOAA (120 instances), (c) RandGauss, (d) magic, (e) elec, (f) checkerboard with continuous drift, (g) SEA (5% noise), and (h) GaussCir.

not necessarily the ideal method to implement concept drift (as it causes catastrophic forgetting [15]), and there are several algorithms that can forget only what is no longer relevant, and retain the still-relevant information, such as the Learn^{++}.NSE algorithm [16;17]. However, since our goal is to evaluate the drift detection mechanism, we use the resetting approach, so that any improvement observed is not due to the innate ability of the learner to learn concept drift, but simply the impact and effect of intervention based on drift detection.

Of the seven datasets, the rotating checkerboard provides the best scenario to determine the effectiveness of HDDDM since the drift occurs at evenly spaced intervals: approximately every 20 time steps, out of 300, the board rotates 0.449 radians, and remains static during the intermediate time stamps. This leads to a total of 15 concepts (including time step 1) throughout the experiment. Table II presents the F-measure, sensitivity and specificity of the HDDDM's detections as averages of 10 independent trials. We can compute these quantities, precisely because the locations of the drift points are known for this dataset. In Table II, sensitivity (ω_1) represents the ratio of the number of drifts detected to total number of real drifts that were actually present for ω_1, whereas specificity (ω_2) is the ratio of number of no-drift time steps to total number of no-drift steps in class ω_2. The performance measure in Table II indicates the detection rate across all data. Note that with class label removed, there is no change in checkerboard distribution, and hence there should be no change detected. The numbers are therefore de facto specificity figures for the entire data. Sensitivity cannot be measured on the entire data (when class labels are removed) since there are no actual drifts in such data.

TABLE II. F-MEASURE, SENSITIVITY AND SPECIFICITY MEASURES ON THE ROTATING CHECKERBOARD DATASET AVERAGED OF 10 INDEPENDENT TRIALS. TRUE POSITIVES CORRESPOND TO THE TRUE POINTS OF CHANGE IN THE DATA.

Parameter →	γ				α	
Measure ↓	0.5	1.0	1.5	2.0	0.05	0.1
F-measure (ω_1)	0.80	0.84	0.78	0.64	0.79	0.82
Sensitivity (ω_1)	1.0	0.97	0.81	0.61	0.98	1.0
Specificity (ω_1)	0.97	0.98	0.98	0.99	0.97	0.97
F-measure (ω_2)	0.79	0.81	0.78	0.64	0.82	0.80
Sensitivity (ω_2)	0.99	0.94	0.86	0.58	0.97	0.98
Specificity (ω_2)	0.97	0.98	0.98	0.98	0.89	0.97
Performance	0.81	0.87	0.91	0.92	0.85	0.82

Table II also shows the effect of the variation of γ and σ on the checkerboard dataset. We observe that a smaller γ is better for sensitivity (and F-measure), whereas a larger γ is better for specificity – not a surprising outcome – with $\gamma = 1$ providing a good compromise. The standard deviation implementation of HDDDM generally maintains a higher performance than the t-statistic implementation; however the latter is more tolerant to changes in its parameter (of α values). We should note that by *performance* we are referring to the performance of the HDDDM algorithm and *not* the performance of the naïve Bayes classifier.

Fig. 9 and 10 display the location of drift detection on ω_1 and ω_2 from the rotating checkerboard dataset, as an additional figure of merit. Recall that this dataset experiences a change every ~20 time steps. There are 15 distinct points of drift (including first time step) in the checkerboard dataset each indicated by the vertical gridlines, whereas each horizontal gridline indicates a different selection of the free parameters of the HDDDM algorithm.

These plots provide a graphical display on the algorithm's ability to detect the changes. Every marker that coincides with the vertical lines is a correct detection of drift. Every marker that falls off a vertical grid is a false alarm of a non-existing drift, whereas every missing marker on a vertical grid is a missed detection of an actual drift. We observe from these plots that the algorithm was able to make the correct call in vast majority of the cases. However, a few cases are worth further discussion: for example, there are certain times when

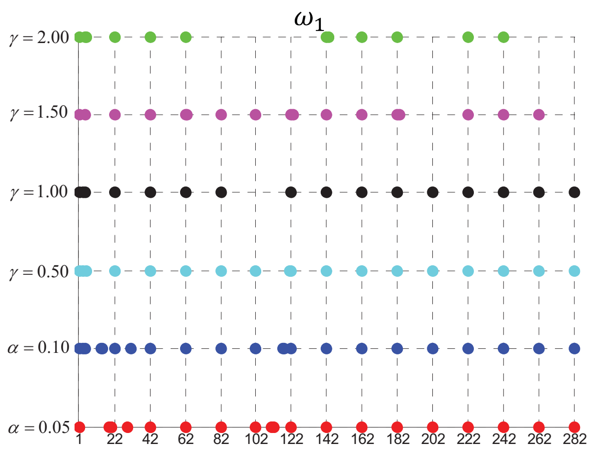

Fig. 9. Location of drift points for ω_1 as detected using the HDDDM on the rotating checkerboard problem with abrupt drift where the x-axis indicates the location of the change and the y-axis is a variation of a parameter.

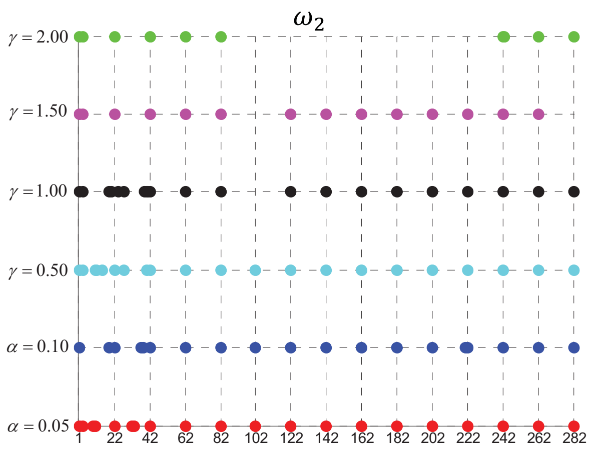

Fig. 10.Location of drift points for ω_2 as detected using the HDDDM on the rotating checkerboard problem with abrupt drift where the x-axis indicates the location of the change and the y-axis is a variation of a parameter.

drift is detected in one of the classes but not the other, even though the drift existed on both classes. Consider the situation for $\gamma = 1.5$, where a change was detected in ω_1 at time stamp 102, while drift was not detected in ω_2. However, the HDDDM algorithm will correctly detect the change in the data in this case, because our implementation decides on a change when drift is detected in either one of the classes. This approach will accommodate all cases where the drift is detected in at least one of the classes. The drawback, however, is a potential increase in false alarm rate: if the algorithm incorrectly detects drift for any of the classes, it will indicate that the drift exists, even though it actually may not. We plan to address this issue in our future work. We also observe that the HDDDM implemented with $\alpha = 0.05$ is quite sensitive and detects change rather often compared to the other implementations of HDDDM.

We should also note that a table similar to Table II, or figures similar to Fig. 9 and 10 cannot be generated for other databases, either because the drift is continuous, or the exact locations of the drift are actually not known.

V. Conclusions

We have presented a drift detection algorithm, inspired in part by [10], that relies only on the raw data features to estimate whether drift is present in a supervised incremental learning scenario. This approach utilizes the Hellinger distance as a measure to infer whether drift is present between two batches of training data using an adaptive threshold. The adaptive threshold was analyzed using a standard deviation and t-statistic approach. This threshold is computed based on the information obtained from the divergence between training distributions. Preliminary results show that the Hellinger distance drift detection method (HDDDM) presented in this paper can improve the performance of an incremental learning algorithm by resetting the classifier when a change has been detected. The primary goal of this preliminary effort was to see whether the algorithm can detect a drift accurately, and whether an intervention based on such detection would benefit the overall classifier performance. The answers to both questions appear to be affirmative. Since our goal was not necessarily to obtain the best concept drift classifier for a given dataset, the actual classifier used in the algorithm was not optimized. The drift detection method proposed in this paper can then be employed with any active concept drift algorithm. Note that the drift detection in HDDDM is not based on classifier error, hence HDDDM is a classifier independent approach and can be used with other supervised learning algorithms.

Future work will include the integration of other (non-parametric) statistical tests along with the Hellinger distance as well as other distance measures. Comparison to other drift detection algorithms, as well as addressing the false alarm issue in cases when the algorithm detects a (non-existing) drift in only one of the classes, are also within the scope of our future work.

References

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

[2] C. Alippi, G. Boracchi, and M. Roveri, "Just in time classifiers: managing the slow drift case," *International Joint Conference on Neural Networks*, pp. 114-120, 2009.

[3] L. I. Kuncheva, "Using control charts for detecting concept change in streaming data," School of Computer Science, Bangor University,BCS-TR-001-2009, 2009.

[4] B. Manly and D. MacKenzie, "A cumulative sum type of method for environmental monitoring," *Environmetrics*, vol. 11, pp. 151-166, 2000.

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

[6] 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.

[7] C. Alippi, G. Boracchi, and M. Roveri, "Change Detection Tests Using the ICI Rule," *World Congress of Computational Intelligence*, pp. 1190-1196, 2010.

[8] M. Baena-Garcia, J. del Campo-vila, R. Fidalgo, and A. Bifet, "Early drift detection method," *4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, pp. 77-86, 2006.

[9] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with Drift Detection," *SBIA Brazilian Symposium on Artificial Intelligence*, vol. 3741, pp. 286-295, 2004.

[10] D. A. Cieslak and N. V. Chawla, "A Framework for Monitoring a Classifiers' Performance: When and Why Failures Occurs?," *Knowledge and Information Systems*, vol. 18, no. 1, pp. 83-108, 2009.

[11] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 31, no. 4, pp. 497-508, 2001.

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

[13] 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*, pp. 377-382, 2001.

[14] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998.

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

[16] M. Muhlbaier and R. Polikar, "An Ensemble Approach for Incremental Learning in Nonstationary Environments," *7th International Workshop on Multiple Classifier Systems*in Lecture Notes in Computer Science, vol. 4472, Springer, pp. 490-500, 2007.

[17] R. Elwell and R. Polikar, "Incremental Learning in Nonstationary Environments with Controlled Forgetting," *International Joint Conference on Neural Networks*, pp. 771-778, 2009.