Discounted Expert Weighting for Concept Drift

Gregory Ditzler and Gail Rosen
Department of Electrical & Computer Engineering
Drexel University
Philadelphia, PA 19148 USA
Email: gregory.ditzler@gmail.com, gailr@ece.drexel.edu

Robi Polikar
Department of Electrical & Computer Engineering
Rowan University
Glassboro, NJ 08028 USA
Email: polikar@rowan.edu

***Abstract*—Multiple expert systems (MES) have been widely used in machine learning because of their inherent ability to decrease variance and improve generalization performance by receiving advice from more than one expert. However, a typical MES explicitly assumes that training and testing data are independent and identically distributed (iid), which, unfortunately, is often violated in practice when the probability distribution generating the data changes with time. One of the key aspects of any MES algorithm deployed in such environments is the decision rule used to combine the decisions of the experts. Many MES algorithms choose adaptive weighting schemes that adjust the weights of a classifier based on its loss in recent time, or use an average of the experts probabilities. However, in a stochastic setting where the loss of an expert is uncertain at a future point in time, which combiner method is the most reliable? In this work, we show that non-uniform weighting experts can provide a stable upper bound on loss compared to techniques such as a follow–the–leader or uniform methodology. Several well-studied MES approaches are tested on a variety of real-world data sets to support and demonstrate the theory.**

***Index Terms*—concept drift; nonstationary environments; multiple expert systems**

I. Introduction

The traditional probably approximately correct (PAC) learning model aims to develop a hypothesis that can learn a function with high probability such that the hypothesis has low generalization error [1]. While the standard PAC model has shown great success in machine learning, particularly within an ensemble of classifiers setting [2], [3], the bounds and guarantees provided by the model do not generalize when on data sampled from multiple sources [4], or when a source that changes over time [5]. One of the assumptions that a PAC model makes is that data are sampled iid from a fixed – albeit unknown – probability distribution, which we denote as $\mathcal{D}$. Unfortunately, the iid assumption is violated far too often in practice because many real-world data sources are nonstationary (e.g., network usage [6], spam [7], weather [8], [9], and electricity pricing [10]). This phenomenon of learning from a nonstationary source commonly referred to as concept drift [11], or simply learning in nonstationary environments. The drift in data can be real or virtual in nature. Real drift is a change in $p(\omega_c|\mathbf{x})$, while virtual drift is an observed change – 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 prediction on new data quite difficult.

Our analysis assumes that data are learned in a streaming fashion using a multiple expert system (MES) with a base expert that is capable of incremental learning (i.e., access to old data are not required to make predictions on new data). At discrete time intervals, denoted by time t, a batch of m labeled data instances are accessed from a probability distribution $\mathcal{D}_t$. An expert, h_t, is generated on the m instances of data sampled from $\mathcal{D}_t$ and the new expert is used with the other experts h_k $\forall k \in \{1, \ldots, t-1\} = [t-1]$ to make predictions. Each expert is assigned a weight for their decision. At a later time, $t = \hat{t}$, a batch of unlabeled data are obtained from a probability distribution $\mathcal{D}_{\hat{t}}$; however, $\mathcal{D}_t \neq \mathcal{D}_{\hat{t}}$ for $t \neq \hat{t}$. The unlabeled field / target data only become available after time t and the data must be classified by the MES. In this work, we denoted the "target" distribution by $\mathcal{D}_T$ (i.e., $\hat{t} = T$).

Given the scenario just described, it would be desirable to determine the advantages and disadvantages of an expert combiner scheme (e.g., weighted voting, unweighted voting, etc) in such a classification scenario. Furthermore, we are also interested in the expected loss of the combiner on the target distribution $\mathcal{D}_T$, and in this work we obtain a simple upper bound on the loss of an MES on a target distribution. Specifically, we examine the effect of a stochastic process governing the loss of experts over time and the effect that has on the loss bound. We also examine the differences between weighting methods on synthetic and real–world data sets.

II. Related Work

A variety of methods have been proposed to handle concept drift including single classifier methods, drift detection, and MES based solutions to name a few. Typically, concept drift algorithms can be categorized as active or passive based on the approach they use to cope with concept drift. 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. For example, `ADWIN` is an adaptive windowing method for detecting changes in a data stream using a statistical hypothesis test [12]. A passive approach simply assumes that some amount of drift

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.

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

may be present in the data and adjusts algorithm parameters whenever new data become available. Learn^{++}.NSE is an example of an MES that assumes drift may happen at any time and hence does not implement explicit drift detection for concept drift, rather concept drift is addressed in a discounted loss for determining expert weights [5]. In this work, we are focusing on passive drift algorithms for handling concept drift in a data stream and how different weighting mechanisms affect an MES loss in a dynamic (nonstationary / drifting) environment.

Multiple expert systems (MES), also known as multiple classifier systems or ensembles, are popular for learning concept drift because of their simple, yet well-founded theory based on variance reduction, and their ability to strike a meaningful balance between stability and plasticity. 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. There are several concept drift MES approaches that are conceptually similar to one another; however, their differences lie in how expert weights are determined and how expert pruning is implemented (if any). For example, Learn^{++}.NSE generates a classifier with each new data set and uses a dynamic weighting strategy that features a weighted sum of each classifier's time–adjusted errors over current and recent environments [5]. Hence, the weights are determined through a discounted loss, which emphasizes recent losses more heavily than old losses. Street and Kim's streaming ensemble algorithm (SEA) adds new experts to the ensemble when new data become available just as with Learn^{++}.NSE [13]; however, unlike Learn^{++}.NSE, SEA uses an upper limit on the ensemble size. SEA prunes experts based on a measure of quality of each expert and generally uses weights proportional to the 1–0 loss or a uniform set of weights. Both Learn^{++}.NSE and SEA share the same approach to adding classifiers; however, expert weights are determines through very different mechanisms. Other similar methods include REA [14], UCB [15], WEA/TRANSE [9], [16], and DDD [17]. In this work, we limit the breadth of our study to examining how loss and the weighting mechanism affects the final loss of an MES on a data stream with concept drift.

III. Formulation and Notation for Learning an MES on a Data Stream

A. The Learning Process

In the following sections, we examine a general framework for a MES that receives batches of data $\mathcal{S}_t$ at time t , sampled from the probability distribution $\mathcal{D}_t$ (Fig. 1). An expert, $h_t \in \mathcal{H}$, is generated on m instances in $\mathcal{S}_t$, and the weights, $w_{t,t}$, of the experts are updated using any suitable weighting method for concept drift algorithms with $\mathcal{S}_t$. The MES decision is $\mathbf{w}_t^\mathsf{T}\mathbf{h}$, where $\mathbf{w}_t = [w_{1,t}, \ldots, w_{t,t}]^\mathsf{T}$ and $\mathbf{h}_t = [h_1, \ldots, h_t]^\mathsf{T}$. We assume that $h_k \in \mathbb{R}$ or $\{\pm 1\}$, $\mathbb{1}^\mathsf{T}\mathbf{w}_t = 1$, and $H = \mathbf{w}_t^\mathsf{T}\mathbf{h}_t$. The label predicted by the MES is $\hat{y} = \text{sign}(\mathbf{w}_t^\mathsf{T}\mathbf{h}_t)$. Note that $\mathbf{w}_t$ forms a convex combination of the experts outputs. Unlabeled data from $\mathcal{D}_{t+1}$ are then received at time $t+1$ and

Input data sets $\mathcal{S}_t$ sampled from distribution $\mathcal{D}_t$
for $t = 1, 2, \ldots$
1) Learn h_t from $\mathcal{S}_t$
2) Update weights w_k $\forall k \in [t]$
3) Predict on unlabeled data sampled from $\mathcal{D}_{t+1}$
4) Receive labels for the data in step 3 and measure expected loss $\mathbb{E}_{t+1}[\ell(H, Y_{t+1})]$

Fig. 1. General implementation of a MES algorithm for concept drift that recieves batches of data at discrete time intervals.

the MES predicts on this unlabeled data. At a later point in time, labels become available and a loss based on a function $\ell(\cdot, \cdot)$.

The aforementioned scenario for learning is quite common in many areas that generate such nonstationary data, and many MES use a similar methodology to Fig. 1 for learning and evaluating loss. As an example, consider forecasting whether stock will rise or fall by the end of the day. Clearly, if the stock price were to increase, it would be desirable to keep the stock (i.e., its investors make money); however, if the stock were to lose value than it should be sold, otherwise its investors incur a loss. It is only after the market has closed that we learn whether the decision to keep or sell the stock has paid off. The problem of forecasting on market prices fits into the generic algorithm described in Fig. 1. In the following sections, we examine the affect that the selection of w_k has on the MES and, in particular, for stochastic data streams, where we cannot be certain about the relationship between $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$.

B. Weighting Experts

There are several ways that the weights, $w_{k,t}$, for an MES can be selected. Ideally, the selection of the weight produces a low generalization loss on future unlabeled data. The simplest approach would be to set $w_{k,t} = \frac{1}{t}$, which is a simple majority vote as suggested by Gao et al. for concept drift [15]. Another weight selection method, known as *follow–the–leader*, sets the weight of the classifier with the smallest loss to $w_{k^*,t} = 1$ with all other experts receiving zero weight. More formally,

$$w_{k,t} = \begin{cases} 1, & k = \arg\min_{k' \in [t]} \ell(h_{k'}, y_t) \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The follow–the–leader method can be quite robust so long as concept drift is not present in the data stream. If concept drift is present, we expect the bound on the loss of the ensemble to be less reliable (i.e., high variance in the bound). However, the follow–the–leader strategy is typically not used in data streams with concept drift when there are multiple experts at our dispense for classification of unlabeled data. Another weighting method is the exponential forecaster, which was popularized by the weighted majority algorithm [3] and heavily analyzed in [18]. In this case, the experts weight accounts for a cumulative loss of each expert, where the weights are now defined as,

$$w_{k,t} = \frac{w_{k,t-1} e^{-\eta \ell(h_k, y_t)}}{\sum_{j=1}^{t} w_{j,t-1} e^{-\eta \ell(h_j, y_t)}} \tag{2}$$

where η is a smoothing parameter. One could use a discounted loss as well for the exponential weighting scheme. A very thorough theoretical analysis of exponential forecaster can be found in [18]. Learn^{++}.NSE's weighting mechanism is a fundamentally different mechanism, which – along with the others mentioned above – we examine in this work. Learn^{++}.NSE uses a discounted loss that applies a normalized logistic sigmoid to an expert's losses over time. Thus, older losses carry lower weight than more recent losses. The final form of Learn^{++}.NSE's weights are that of Adaboost's,

$$w_{k,t} = \log \frac{1 - \hat{\ell}_t(h_k)}{\hat{\ell}_t(h_k)} \tag{3}$$

where $\hat{\ell}(h_k)$ is the discounted loss of expert h_k.

IV. Analysis & Methodology

Forming an upper bound on the loss of a multiple expert system (MES), or any expert in a stationary or nonstationary environment for that matter, is an important aspect of analyzing a classification / forecasting system because it allows for inference on the loss of the expert on unseen data. However, we must be careful in our formulation of the bound because of the training / testing data being sampled from different probability distributions. That is, an error bound such as that of Adaboost's is meaningless because it only applies to the training loss and does not infer the expected loss on data not sampled from the training source [2]. A typical loss bound is only valid on a fixed source, whereas we are considering the phenomenon of concept drift, which occurs when the data sources are changing over time. In this section, we seek to find similar bounds for an MES that have experts each trained on a possibly different probability distribution and tested on some target distribution that changes with time.

A. Loss Functions

The loss function plays an important role in the design and mathematical analysis of any classifier, MES included. In this work we assume that the loss function is convex in its first argument. The selection of the loss function is typically a free parameter to the user, though some loss functions have become the standard (e.g., Wang & Zhou provide a discussion of the advantages and disadvantages of MSE for signal processing in [19]):

$$\ell(h, f) = \begin{cases} \max(0, 1 - h \cdot f), & \text{hinge loss} \\ |h - f|, & \text{absolute loss} \\ \log(1 + \exp\{-h \cdot f\}), & \text{log loss} \\ (h - f)^2, & \text{square loss} \\ \begin{cases} 1 & \text{if } h = f \\ 0 & \text{otherwise} \end{cases}, & \text{1–0 loss} \end{cases}$$

All of these loss functions are convex in their first argument with the exception of the 1–0 loss function, which typically complicates the optimization process because it is non-convex. Fig. 2 shows a comparison of the loss functions discussed in this section.

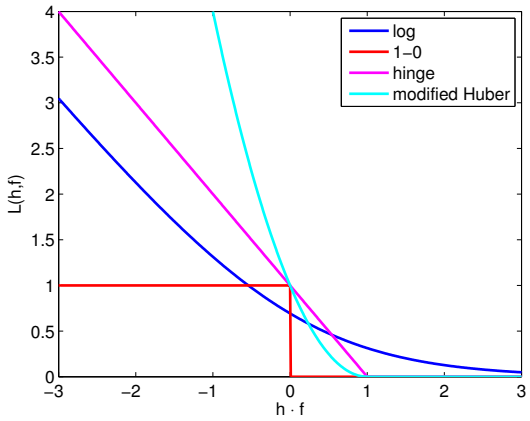

Fig. 2. Standard loss functions used throughout machine learning. Log-loss and hinge loss are convex; however, 1–0 is not convex.

A discounted loss is one that weights recent losses more heavily than losses that occurred a long time ago. As stated earlier, Learn^{++}.NSE is such an algorithm that uses discounted loss to form weights of experts. Mathematically, discounted loss is defined as,

$$\hat{\ell}_t(h_j) = \sum_{k=1}^{t} \rho_{t-k} \ell_k(h_j) \tag{4}$$

where ρ_t are discount factors (i.e., $\rho_0 \geq \rho_1 \geq \ldots \geq \rho_t$). For Learn^{++}.NSE, the ρ factors form a logistic sigmoid. Discounted loss can be quite useful for nonstationary data streams because the discount terms act as forgetting factors, which have been analyzed with algorithms such as Learn^{++}.NSE [20]. Another popular discount function is the exponentially weighted moving average. We implement our exponential forecasters, which uses Eq. (2) to compute the weights, with the exponentially weighted moving average applied to the loss as a discount function.

B. General Bound for Multiple Expert Loss

Let H be the decision / hypothesis that the MES makes given the advice from experts h_k for $\forall k \in [T]$. The MES decision is formed by a convex combination of the expert decisions, that is

$$H = \sum_{k=1}^{T} w_k h_k \tag{5}$$

where the weights are subject to $w_k \geq 0$ and $\sum_k w_k = 1$. Without loss of generality we write $H = H(\mathbf{x})$ and $h_k = h_k(\mathbf{x})$ to denote the prediction given by an MES and expert's prediction on instance $\mathbf{x}$, respectively. Assume that h_k was trained on some distribution $\mathcal{D}_k$, which may different than $\mathcal{D}_{\hat{t}}$ $\forall \hat{t} \neq k$. Let $\ell(f, g)$ be a convex loss function, which must at least be convex in terms of the first argument. The loss of the

MES on the target function on a single example $\mathbf{x}$ is given by

$$\ell(H, f_T) = \ell\left(\sum_{k=1}^{t} w_{k,t} h_k, f_T\right) \leq \sum_{k=1}^{t} w_{k,t}\ell(h_k, f_T)$$

where the last step is a result of Jensen's inequality and the assumption that $\ell(\cdot,\cdot)$ is convex at least in its first argument. The above inequality is only for the loss on a single example from the target distribution $\mathcal{D}_T$. Ideally, we would want to examine the expected value of the MES loss on the target probability distribution. Therefore, we compute the statistical expectation of $\ell(H, f_T)$ over the target distribution $\mathcal{D}_T$,

$$\mathbb{E}_T[\ell(H, f_T)] \leq \mathbb{E}_T\left[\sum_{k=1}^{t} w_{k,t}\ell(h_k, f_T)\right] = \sum_{k=1}^{t} w_{k,t}\mathbb{E}_T[\ell(h_k, f_T)] \quad (6)$$

where $\mathbb{E}_T[\ell(h_k, f_T)]$ denotes the statistical expectation of a function $\ell(h_k, f_T)$ over the target distribution $\mathcal{D}_T$. Note that $\mathcal{D}_T$ is a probability distribution over X, or more formally $X \sim \mathcal{D}_T$.

There are several points that we can make about Eq. (6). First, it is an upper bound for the MES loss on a target (testing) data set – albeit a little uninformative. Second, by convexity, the loss is less than that of worst performing expert and lower bounded by the loss of the best performing expert on the target. Third, the bound is typically not computable because we do not have access to f_T; however, we simulate the results as presented in Section V. If we were to have access to f_T we could use the f_T directly to minimize the expected loss on the target distribution. We should re-emphasize the fact that we do not have access to the labeled data from $\mathcal{D}_T$ at time $t = T$.

C. Designing Weights for MES

So far, we have observed from Eq. (6) that the weights of the experts in the ensemble play a crucial role in the upper bound on the loss of the ensemble. Therefore, we seek to observe the effects of the loss function in conjunction with the weighting mechanism and the effect of the weighting mechanism when the loss of an expert is uncertain at future time stamps. To do so, we compare a variety of widely used expert weighting methods and loss functions from Section III-B and IV-A, respectively.

V. Experimental Results

In this section we present a simulation study of the analytical findings discussed in Section IV as well as results on several real-world data sets collected from UCI as well as other sources. For fairness of comparison, we have used the default parameters as recommended by the authors of the respective classification algorithms. Our primary goal is to: (*i*) examine the effects of expert weight selection when a classifiers loss can be viewed as a stochastic process, (*ii*) examine the effect of expert weight selection on real–world data streams, and (*iii*) observe the relationship between the simulations and real–world data streams. Section V-B presents the results from the simulation study and Section V-C highlights the results from the real-world data sets.

TABLE I
Real-world data sets used in the study of losses for several benchmark MES algorithms for concept drift.

Data set	Cardinality	Features	Batch Size
NOAA (rain)	18159	8	100
Elec. Price (elec2)	27549	5	200
german	1000	20	100
ringworm	7400	20	250
splice	3175	60	100
twonorm	7400	20	100
waveform (wave)	5000	21	200
chess	533	6	35
Luxemburg (lux)	1901	20	50

1: **Input.** $\ell_t(h_k)$ is the state of a random variable of the Markov chain for expert k at time t and $U \sim \mathcal{U}(0,1)$ is a uniform random variable in the interval $[0,1]$.
Def. Set the transitions to $\ell_{t+1}(h_k) \in \{\ell_t(h_k) - 0.05, \ldots, \ell_t(h_k), \ldots, \ell_t(h_k) + 0.05\}$, where the transition from state i to j is uniform over all valid transitions.
2: **for** $t = 1, \ldots, T$ **do**
3: Generate initial loss of classifier $\ell_t(h_t) \sim \mathcal{U}(0, 0.15)$
4: **for** $k = 1 : t-1$ **do**
5: Simulate a one step transition for $\ell_t(h_k)$ given $\ell_{t-1}(h_k)$
6: **end for**
7: **end for**
8: **Output.** Loss matrix $\ell_t(h_k)$

Fig. 3. Simulation study pseudo code for generating a "expert's" loss as a Markov chain with reflecting boundaries.

A. Data Set Descriptions

The data sets, along with their cardinality, number of features and batch size can be found in Table I. The Luxembourg data set was constructed using time series European social survey data [6], [21]. Each observation is an individual and the features are answers to the survey questionnaire. The labels indicate high or low internet usage. The chess data set was constructed using time series data obtained from chess.com portal [22]. The data consists of game records of one player over a period from 2007 December to 2010 March. A player has a rating, which changes depending on his/her results achieved (the higher the rating, the stronger the player is). The weather data set is a subset of the NOAA data that was processed and released as a concept drift dataset [5][1]. 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 Electricity

[1]Link to data: http://users.rowan.edu/∼polikar/research/nie_data/

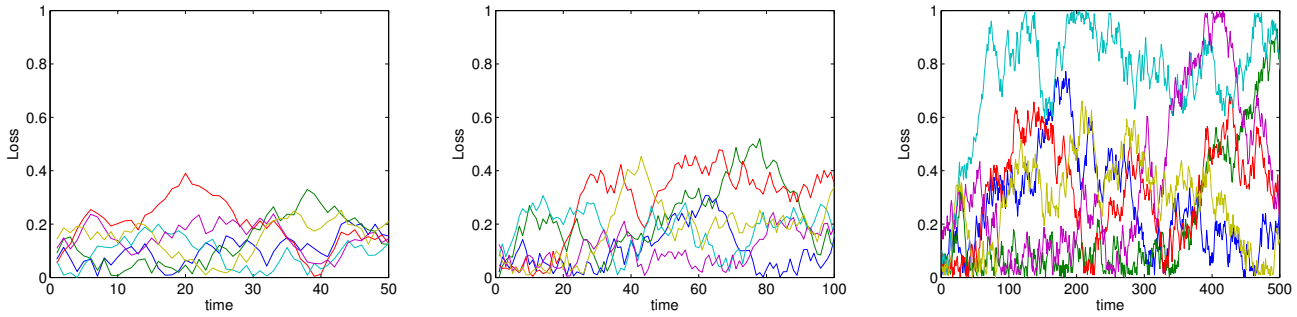

Fig. 4. Simulation of six classifier's loss as a Markov chain with reflecting boundaries as described in Fig. 3. (**left**) Simulation over 50 time stamps, (**middle**) Simulation over 100 time stamps, and (**right**) Simulation over 500 time stamps. For the simulation study we constrain $\ell(H, Y)$ to be in the interval $[0, 1]$.

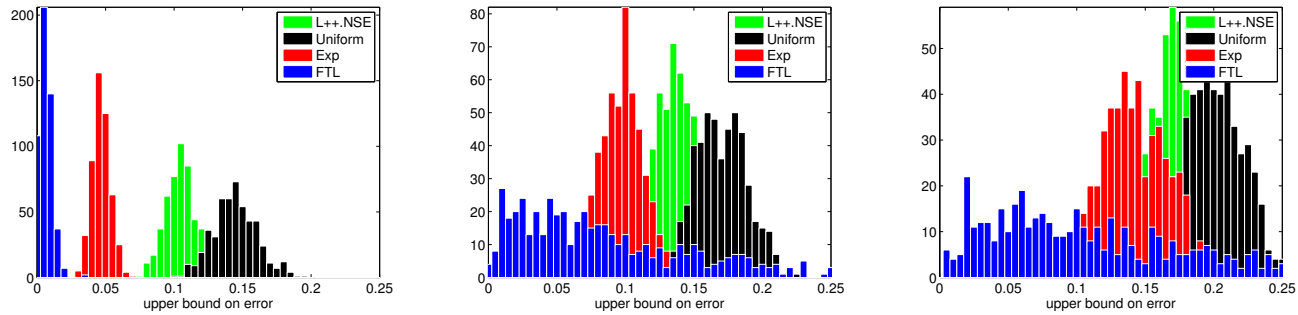

Fig. 5. Distribution of the upper loss bound for exponential weighting (red), Learn^{++}.NSE weighting (green), average weighting (black), and follow–the–leader (blue). (**left**) upper bound on loss at time T (i.e., no bias between the training & testing probability distributions), (**middle**) upper bound on loss at time $T + 10$, and (**right**) upper bound on loss at time $T + 25$. The histograms of the loss bounds are generated using 500 Monte-Carlo simulations of a classifiers loss based on the loss simulation method in Fig. 3.

Pricing dataset (elec2) was first introduced by Harries and has also been used as a benchmark for concept drift problems [10]. The dataset provides time and demand fluctuations in the price of electricity in New South Wales, Australia. All of the aforementioned data sets have some amount of drift due to the environment in which they were collected. The remainder of the data sets were collected from the UCI machine learning database [23] and nature of drift in these datasets is unknown.

B. Stochastic Loss Simulation Study

We begin with a simulation study where the loss of an expert can be viewed as a random variable. In this simulation the loss of an expert can be modeled as a random walk of a Markov chain with reflecting boundaries in $[0, 1]$, where we have uniformly discretized the loss [24]. Hence, the possible outcomes for the loss can be viewed as being a random walk on the set $\left\{\frac{1}{100}, \frac{2}{100}, \ldots, \frac{100}{100}\right\}$ (assuming step sizes of $\frac{1}{100}$).

A summary of the simulation procedure can be found in Fig. 3. We simulate an expert's loss at each time stamp, and a "new expert" is simulated at each time stamp by initializing a new chain. The process begins by generating an initial loss for expert t at time t, $\ell_t(h_t)$ (which is the state of the Markov chain). The initial loss is considered to be a uniform random variable in the interval $[0, 0.15]$. Then, for all existing experts (i.e., h_k for $k \in [t]$), we simulate a one step transition in the chain for the loss of each expert, $\ell_t(h_k)$. That is, each expert's loss is modeled as a Markov chain and is independent of any other expert's loss. The state transitions are defined as $\ell_{t+1}(h_k) \in \{\ell_t(h_k) - 0.05, \ldots, \ell_t(h_k), \ldots, \ell_t(h_k) + 0.05\}$ so long as state $\ell_t(h_k) + m$ is a valid state (invalid states have probability of zero and boundaries are reflective). All transition probabilities are equally likely in our model. We continue this expert loss simulation process T time stamps and return all the observed losses for all experts.

Computing the upper bound on loss at time $t = T$ is not sufficient for us to test MES effectiveness in stochastic environments, because weights computed at time T will be sub–optimal for time $T+m$. Therefore, we compute the weights for each expert at time T and find the m–step transition for each expert's loss. We let the m–step loss the expected loss in Eq. (6) (right side of the inequation); however, the weights from time T are used in the calculation of the bound. Given the m–step loss for all experts and the weights, we can compute the upper bound directly. The simulation process 250 times and examine the distribution of the upper bound on the loss of the MES for exponential weighting, Learn^{++}.NSE weighting, average weighting, and follow–the–leader (FTL).

Fig. 4 shows the simulated loss of six different expert's over 50, 100 and 500 time stamps. The loss simulation process is on par with the methodology described in [15] where an expert's loss at a future time cannot be precisely predicted due to an underlying stochastic process. In the following section

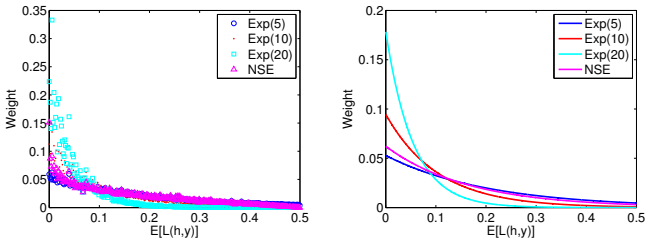

Fig. 6. Weights plotted as a function of loss assuming that $\ell(\cdot,\cdot) \in [0, \frac{1}{2}]$. (**left**) classifier voting weight when 10 classifiers loss are viewed as a uniform random variable in the interval $[0, \frac{1}{2}]$. (**right**) exponential regression of the weight on the *left*.

we examine the upper bound on MES loss for a variety of different voting mechanisms including that of uniform weighting, exponential weighting, Learn^{++}.NSE weighing, and a follow–the–leader strategy.

Fig. 5 shows the distribution of the upper bounds on MES loss for the simulations. There are several key observations that we can make from this simulation. First, the FTL approach provides the lowest upper bound when MES is predicting on data sampled from time T (i.e., there is no change in any of the expert's loss since the leader was assigned). As the m-step transitions occur in the simulation, and the FTL's upper bound on loss becomes less reliable because of change in the leader's loss over the m-step transition. This high variance in the bound suggests that the FTL approach, while simple in its weight assignment, provides much more variance in its expected upper bound as the MES faces more of a "forecasting" problem than "classification" problem. Second, with an appropriate selection of $\eta_{k,t}$ the exponentially weighted MES provides a lower upper bound on the loss compared to Learn^{++}.NSE and average weighting. This lower upper bound is due to experts with a lower loss being weighted heavier than Learn^{++}.NSE would weight the same experts (refer to Fig. 6). Note that the original implementation of Learn^{++}.NSE did not normalize the weights of the experts [5]. We normalized the weights to compare Learn^{++}.NSE's to the exponential weighting method. Third, MES implementations with weighted voting, where the weights are derived from a discounted loss, provide lower upper bounds on MES loss than the average weighting.

C. Results on Real-World Data Sets

The real–world data sets are split into batches using the batch size specified in Table I. We use a test–then–train strategy to evaluate the MES on the data stream. That is, the MES begins by testing on a data set whose labels are not available to the algorithm. Then, the labels become available, and the data set is used for training at the next time stamp. A linear classifier is trained as the base–expert at each time stamp, and the classifiers are combined using the four different weighting schemes discussed in the previous section. The loss of each method is measured as the loss that was used to determined the expert voting weights. For example, if the logistic loss was used to determine the expert weights than the logistic loss is reported as a measurement of the quality of the ensemble decision.

Our implementation of Learn^{++}.NSE is slightly different than the implementation of the algorithm as described by Elwell [5]. Specifically, only the weighting mechanism of Learn^{++}.NSE is used as the weight update for the algorithm in Fig. 1. This is because, we are primarily interested in examining the discounted loss strategy used by Learn^{++}.NSE. We refer to the Learn^{++}.NSE weighting strategy as used in this work as $\mathsf{L}^{++}.\mathsf{NSE}$, exponential as Exp, majority voting as $\mathsf{Uniform}$, and follow–the–leader as FTL. The slope and cutoff for the Learn^{++}.NSE sigmoid are given by $\frac{1}{2}$ and 10 , as recommended by its authors in [5], respectively. The exponential weighting method is implemented with an exponentially weighted discounted loss with a smoothing parameter of $\frac{1}{2}$.

The results of the different MES algorithms on the real–world data sets are shown in Table II. The results are divided into three sections. One section for each loss function tested (i.e., 1–0, logistic and mean–squared error). First, we observe that uniform weights do not provide superior results over the non–uniform weighting methods such as that of Learn^{++}.NSE or exponential weighting. In fact, uniform weighting, in terms of averaged rank, never outperforms Learn^{++}.NSE's discounted loss scheme. Furthermore, with the 1–0 loss function, the uniform weighting methods cannot beat the follow–the–leader forecasting strategy when examined across all data sets. These results coincide with the results of the stochastic simulation experiment in the previous section. Second, the discounted loss used by Learn^{++}.NSE is quite robust and generally performing near the top for all loss functions tested in this work. We emphasize that the results are obtained using the default parameters (i.e., no parameter tuning was performed). Third, even though the follow–the–leader prediction strategy has a high variance distribution on the upper bound for loss in the simulation study, the follow–the–leader strategy is not a top performer across all the data sets.

VI. Conclusion

In this work, we have examined the role that an expert's weight plays in the development of a loss bound of the ensemble decision on a target distribution. We are interested in the effects the weighting schemes, using the basic MES in Fig. 1, will have on an MES when tested on real–world and synthetic data sets. We strictly examine the effects of the weighting methods that an MES uses, not any specific algorithm such as Learn^{++}.NSE. The variation between the different weighting mechanisms were tested by examining a simulation study and real–world data sets. We find that the distribution of the upper bound on the loss of the follow–the–leader contains a higher variance than weighting methods such as exponential or the Learn^{++}.NSE discounted loss scheme. Furthermore, we find that the weighting voting schemes, such as Learn^{++}.NSE and exponential weighting, provide better overall ranks than uniform voting or the follow–the–leader strategy. This result suggests that using an ensemble of experts is more advantageous than strictly using a single expert in

TABLE II

AVERAGE LOSS FOR FOUR EXPERT WEIGHING METHODS ON NINE DATA SETS AND FOUR LOSS FUNCTIONS. THE RANK OF EACH ALGORITHM IS REPORTED FOR EACH DATA SET AND LOSS FUNCTION TESTED. AVERAGE RANKS FOR EACH WEIGHTING METHODS ARE REPORTED IN THE RIGHT MOST COLUMN. NOTE, 1–0 LOSS IS THE ONLY LOSS FUNCTION USED THAT IS NOT A CONVEX FUNCTION.

	Data set									
	rain	*elec2*	*german*	*ringworm*	*splice*	*twonorm*	*wave*	*chess*	*lux*	rank
					1–0 loss					
nse	0.218 (3)	0.339 (1)	0.224 (1)	0.232 (2)	0.18 (1)	0.0234 (2)	0.139 (1)	0.329 (3)	0.102 (1)	1.67
avg	0.216 (2)	0.365 (4)	0.259 (4)	0.242 (3)	0.206 (3)	0.0245 (3)	0.152 (3)	0.419 (4)	0.499 (4)	3.33
exp	0.214 (1)	0.354 (3)	0.229 (2)	0.232 (1)	0.184 (2)	0.0231 (1)	0.144 (2)	0.31 (2)	0.485 (3)	1.89
ftl	0.294 (4)	0.341 (2)	0.243 (3)	0.249 (4)	0.271 (4)	0.0387 (4)	0.162 (4)	0.288 (1)	0.13 (2)	3.11
					logistic loss					
nse	0.781 (3)	0.897 (2.5)	0.69 (1.5)	0.747 (1)	0.756 (1.5)	0.489 (1)	0.637 (1)	0.879 (3)	0.992 (2)	1.83
avg	0.781 (2)	0.897 (2.5)	0.69 (1.5)	0.747 (2)	0.756 (1.5)	0.491 (3)	0.637 (2)	0.879 (4)	1.03 (4)	2.50
exp	0.779 (1)	0.892 (1)	0.691 (3)	0.748 (3)	0.756 (3)	0.491 (2)	0.637 (3)	0.873 (2)	0.998 (3)	2.33
ftl	0.87 (4)	0.938 (4)	0.738 (4)	0.795 (4)	0.824 (4)	0.502 (4)	0.665 (4)	0.83 (1)	0.627 (1)	3.33
					mse loss					
nse	0.652 (3)	0.896 (3)	0.716 (1.5)	0.705 (1.5)	0.569 (1.5)	0.0759 (1)	0.454 (3)	0.991 (3)	0.377 (1)	2.06
avg	0.625 (1)	0.892 (2)	0.716 (1.5)	0.705 (1.5)	0.569 (1.5)	0.0768 (3)	0.41 (2)	0.933 (2)	1.47 (4)	2.06
exp	0.63 (2)	0.872 (1)	0.719 (3)	0.707 (3)	0.572 (3)	0.0766 (2)	0.409 (1)	0.918 (1)	1.1 (3)	2.11
ftl	1.18 (4)	1.37 (4)	0.971 (4)	0.997 (4)	1.09 (4)	0.155 (4)	0.646 (4)	1.15 (4)	0.52 (2)	3.78

the decision process. This result has been noticed in previous experiments with concept drift algorithms.

Our current and future work includes deriving a more precise and tighter upper bound on the loss of an MES on a target distribution. Such a framework would allow researchers building MES algorithms to develop loss bounds for their classifiers, and determine how their classifier's loss bound compares to others. Furthermore, developing a computable bound would be highly informative (i.e., no knowledge of $\mathcal{D}_T$ or f_T is required) along with methods to optimize the parameters of the weighting schemes.

REFERENCES

[1] L. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, pp. 1134–1142, 1984.

[2] Y. Freund and R. Shapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[3] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, pp. 212–261, 1994.

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

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

[6] R. Jowell and the Central Coordinating Team, "European social survey 2002/2003; 2004/2005; 2006/2007," tech. rep., London: Centre for Comparative Social Surveys, City University, 2003, 2005, 2007.

[7] S. Delany, P. Cunningham, and A. Tsymbal, "A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering," in *International Conference on Artificial Intelligence*, pp. 340–345, 2006.

[8] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. Shapire, and M. K. Warmuth, "How to use expert advice," *Journal of ACM*, vol. 44, no. 3, pp. 427–485, 1997.

[9] G. Ditzler, G. Rosen, and R. Polikar, "A transductive learning algorithm for nonstationary environments," in *International Joint Conference on Neural Networks*, 2012.

[10] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," tech. rep., The University of South Wales, 1999.

[11] 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.

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

[13] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large scale classification," in *Proceedings to the 7th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 377–382, 2001.

[14] S. Chen and H. He, "Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach," *Evolving Systems*, vol. 2, no. 1, pp. 35–50, 2011.

[15] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 203–208, 2007.

[16] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," in *International Joint Conference on Neural Networks*, 2011.

[17] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge Discovery and Data Engineering*, vol. 24, no. 4, pp. 619–633, 2012.

[18] M. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[19] Z. Wang and A. C. Bovik, "Mean squated error: Love it or leave it?," *Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[20] R. Elwell and R. Polikar, "Incremental learning in nonstationary environments with controlled forgetting," in *International Joint Conference on Neural Networks*, pp. 771–778, 2009.

[21] I. Žliobaitė, "Combining time and space similarity for small size learning under concept drift," in *International Symposium on Methodologies for Intelligent Systems*, vol. 5722, pp. 412–421, 2009.

[22] I. Žliobaitė, "Change with Delayed Labeling: when is it detectable?," in *IEEE International Conference on Data Mining Workshops*, 2010.

[23] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[24] G. Lawler, *Introduction to Stochastic Processes*. Chapman and Hall/CRC, 2006.