An Ensemble Based Incremental Learning Framework for Concept Drift and Class Imbalance

Gregory Ditzler, *Member, IEEE* and Robi Polikar, *Senior Member, IEEE*

***Abstract*—We have recently introduced an incremental learning algorithm, Learn^{++}.NSE, designed to learn in nonstationary environments, and has been shown to provide an attractive solution to a number of concept drift problems under different drift scenarios. However, Learn^{++}.NSE relies on error to weigh the classifiers in the ensemble on the most recent data. For balanced class distributions, this approach works very well, but when faced with imbalanced data, error is no longer an acceptable measure of performance. On the other hand, the well-established SMOTE algorithm can address the class imbalance issue, however, it cannot learn in nonstationary environments. While there is some literature available for learning in nonstationary environments and imbalanced data separately, the combined problem of learning from imbalanced data coming from nonstationary environments is underexplored. Therefore, in this work we propose two modified frameworks for an algorithm that can be used to incrementally learn from imbalanced data coming from a nonstationary environment.**

***Index Terms*—concept drift, imbalanced data, ensemble of classifiers, incremental learning in nonstationary environments**

I. Introduction

Concept drift, associated with learning in nonstationary environments, receives substantially less attention in most classical machine learning literature, particularly if such an environment generates imbalanced class distributions. Concept drift can be defined as a change in the underlying distribution that generates the data used to train a classifier. The problem is that classifiers trained on previously available data may become obsolete. While learning in nonstationary environments and class imbalance has been researched independently and several novel algorithms have been proposed to handle nonstationary concepts or imbalanced data, there has been relatively little work done with the combination of these problems [1-3]. Learning in a nonstationary environment requires that the learner is able to learn from a concept that is changing in time. This change can be real or virtual. Real drift is a change in the likelihoods while a virtual drift is a result of an incomplete representation of the true distribution in the current data. Real and virtual drift may occur at the same time, and it can be difficult to determine which one is occurring and even more difficult to tell if both are occurring at the same time [4].

The main contribution of this work is a supervised ensemble of classifiers based incremental learning algorithm that is designed to work in nonstationary environments, experiencing class imbalance in the data. This framework is based on the Learn^{++}.NSE algorithm; however, the error of each classifier is no longer the contributing factor to the weighting scheme. Following a review of approaches for imbalanced data, nonstationary learning and a combination of the two in described in Section II, we describe the algorithm in Section III, followed by the results on several databases subject to concept drift and class imbalance presented in Section IV. Finally, Section V contains conclusions and final remarks.

Manuscript received January 31, 2010. The manuscript was revised and resubmitted on May 2, 2010. This work was supported by the National Science Foundation under Grant No: ECCS-0926159.

Authors are with the ECE Department at Rowan University and are part of the Signal Processing & Pattern Recognition Lab, Glassboro, NJ, 08028, USA (e-mail: gditzler@ieee.org, polikar@rowan.edu).

II. Background

A. Nonstationary Environments

One of the earliest approaches for classifying data in a nonstationary environment uses a sliding window, whose size is determined by the rate of drift. Therefore, an algorithm that uses an adjustable window typically follows an active approach to drift detection, constantly seeking to detect change as presented in [5-9]. Typically, in such algorithms, there is a drift detection mechanism that updates the current model only when the drift is detected, assuming that the old model (and hence the old data) is no longer relevant. The faster the drift rate, the shorter the window length, with the understanding that older data are becoming increasingly less relevant as the environment is changing. Conversely, the window size grows if the drift is slow or nonexistent with the understanding that the data from several time steps ago may still be relevant and useful for classification purposes. The FLORA family of algorithms was one of the first methods that employed the dynamic window length approach [8]. While this approach is very simple, it does not allow for incremental learning, since incremental learning requires learning the knowledge from the current data and existing model(s), without requiring access to previous data. A passive approach to learning concept drift, on the other hand, simply accepts that a concept drift may or may not have occurred, and updates the model with each incoming batch of the data stream. The algorithms proposed in [1;3;10-12] are all passive algorithms.

Multiple-classifier systems (MCS), or ensembles, have been suggested as an attractive method of learning concept drift in [13], based on their natural ability to obtain a good balance between stability (ability to retain relevant information) and plasticity (ability to acquire new knowledge) [14].

Kolter & Maloof present the dynamic weighted majority (DWM) algorithm in [1] which uses an online learner such

as naïve Bayes, or incremental tree inducer to train an ensemble with the final voting decision obtained by dynamic weighted majority voting. The voting weight of each classifier is set to 1 when created, and is reduced when that classifier misclassifies an instance. Once the classifier's weight falls below a threshold, it is removed from the ensemble.

The Learn^{++}.NSE algorithm, on the other hand, uses a weighted sum of the current and past normalized pseudo errors of each classifier to compute the voting weight [15]. This algorithm applies a sigmoid weighting function to the previous errors of the classifiers. Such an approach increases the weight of a classifier when that classifier obtained low error in recent time steps. Therefore, a classifier that was created many time steps ago may still receive a high voting weight if it performs well on the current environment, and is particularly useful in recurring concepts. Conversely, a classifier can have its voting power virtually removed from the ensemble decision if it is performing poorly in recent time, but this is only until the classifier begins to perform well again (if it ever does). In [16], Learn^{++}.NSE had several different pruning methods applied to the ensemble to limit the ensemble size and monitor the effect of the pruning on the overall performance in a nonstationary environment. While Learn^{++}.NSE works well in a variety of concept drift environments with balanced data, it is not well suited for imbalanced data because the classifier errors is no longer a suitable metric the weighting of classifier.

B. Imbalanced Data

Class imbalance occurs when a dataset does not have an (approximately) equal number of examples from each class, which may be quite severe in some applications [17]. Most approaches for learning from such data are based on under sampling the majority class or oversampling the minority class [18]. While relatively straightforward, each has significant shortcomings: under sampling throws away data from the majority class, whether they are useful or not. Oversampling creates exact replicates of minority class instances, which may cause the classifier to over fit those instances. A more novel approach is followed by SMOTE [2], which modifies the feature space rather than the data space by creating synthetic examples that are located on the line segment connecting two minority neighbors. SMOTE has been shown to improve the classification accuracy of the minority class over other standard approaches. SMOTEBoost was later presented in [19] as an improved alternative combining SMOTE and AdaBoost.M2 so that the f-measure and recall of the ensemble can be increased. More recently, the bagging ensemble variation (BEV) was proposed in [20], which uses a form of bagging that trains classifiers with all the minority class data and subsets of the majority class data.

C. Imbalanced Data with Concept Drift

Recently, a framework for an algorithm that is capable of learning in a nonstationary environment with imbalanced data has been proposed in [11]. The algorithm is based on a bagging framework that trains classifiers (C4.5 or naïve Bayes) on a portion of the majority class that is controlled with a user defined parameter between [0,1] and the total number of minority class instances up until the most recent time step (current + previous positive examples). With each iteration, the minority class instances are saved to be used to train classifiers at the next iteration when a new database is introduced. However, this approach implicitly assumes that the minority class is stationary, which may not be true. Furthermore, the approach cannot be formally considered incremental since it requires access to old data.

In [12], a bagging based approach that uses a similarity measure (such as the Mahalanobis distance) to select previous minority examples that are most similar to the newest dataset was used for learning in a nonstationary environment with class imbalance. In this approach, examples that are irrelevant are effectively discarded from a current training set by employing the Mahalanobis distance. There is also an underlying assumption that the minority data come from a Gaussian distribution. The final ensemble decision is made using majority voting. However, this framework for handling both class imbalance and nonstationary environments is not suited for incremental learning as it also requires access to the previous data. Using such data forces an implicit assumption that the minority class (concept) is stationary, a potentially incorrect assumption, with access to previous data also violating the definition of incremental learning. Our goal in this effort is to propose a framework that can: (1) learn incrementally without access to the previous data, (2) build a set of classifiers that are robust to class imbalance, and (3) learn in a nonstationary environment without relying on error as the weighting metric. Starting with the Learn^{++}.NSE as a stepping stone, we propose an algorithm that meets these criteria.

III. Algorithmic Framework

A. An Overview of the Proposed Approach

Our primary goal is to develop an ensemble of classifiers model that can recognize instances of both the minority and the majority class, whose distributions may be experiencing concept drift. Since Learn^{++}.NSE has been shown to work well under various drift conditions [21], it was chosen as the foundation for its successor, Learn^{++}.NIE (***N**onstationary and **I**mbalanced **E**nvironments*). An alternative is to use SMOTE as a precursor to Learn^{++}.NSE to balance the data distribution prior to nonstationary learning. We describe the former in detail, and refer to individual references of [2;3;16] for the latter, since Learn^{++}.SMOTE is a straightforward concatenation of the two algorithms.

Learn^{++}.NSE creates a new member of the ensemble with each new batch of data, evaluates the ensemble on the current data, creates a weighted average of classifier errors on current and recent environments, and assigns voting weights to each classifier based on age-adjusted weighted errors. The final decision is then obtained as the weighed majority voting of all classifiers.

Learn^{++}.NIE is also presented with batches of data in an

incremental fashion where the current distribution, $p^{(t)}(\boldsymbol{x}, \omega_c)$ may be different than, $p^{(t-1)}(\boldsymbol{x}, \omega_c)$, the distribution from which prior batch of data was drawn. However, two major distinctions separate Learn^{++}.NSE from Learn^{++}.NIE: (1) the new algorithm creates a sub-ensemble of classifiers for each batch of data (as opposed to a single new classifier); and (2) a different metric (not classifier error) is used as an evaluation measure. As mentioned earlier, Learn^{++}.NSE relies primarily on classification error to determine voting weights of the classifiers, which works well in nonstationary environments that have balanced data class distributions. However, error is not a reliable metric in imbalanced datasets; for example, in a dataset in which the minority class constitutes only 1% of the instances, blindly choosing majority class gets 99% overall classification accuracy, but 0% on the minority class, which is usually the more important class. Therefore, we explore using a class-specific weighted error, and chose the metric shown in Eq. 1, for updating classifiers weights.

$$\epsilon_k^{(t)} = \eta\left(1 - r_{k,(+)}^{(t)}\right) + (1-\eta)\left(1 - r_{k,(-)}^{(t)}\right) \qquad (1)$$

where $r_{k,(+)}^{(t)}$ and $r_{k,(-)}^{(t)}$ are the recall of the kth sub-ensemble on the positive (minority) and negative (majority) class at time step t, respectively. This metric rewards a classifier with a higher voting weight, if it has a high recall on both minority and majority classes.

The age-adjusted weighted average of the errors is obtained through a logistic sigmoid, which provides a higher weight to errors on recent environments. The sub-ensembles that are performing well on both classes in recent times are therefore awarded with higher weights. The slope and cutoff of the logistic sigmoid can be controlled based on the predicted rate of drift.

B. Algorithm Description

The Learn^{++}.NIE algorithm, whose pseudo code is shown in Figure 1, is presented with database, $\mathcal{D}^{(t)}$, at time step t. The algorithm is designed to work in a nonstationary environment so if the distribution of $\mathcal{D}^{(t)}$ is $p^{(t)}(\boldsymbol{x}, \omega_c)$, then $p^{(t)}(\boldsymbol{x}, \omega_c)$ need not be the same as $p^{(t-1)}(\boldsymbol{x}, \omega_c)$, the distribution of $\mathcal{D}^{(t-1)}$. Since this is an incremental learning algorithm, access to the previous databases is not required. Therefore, each sub-ensemble must serve as a model for all the data at time step t. A bagging variation method is called to create a small ensemble of (K=3) classifiers (step 1).

Traditional bagging generates classifiers trained on $m' \leq m$ randomly sampled examples from the database. The form of bagging used in the proposed framework trains a classifier on all of the minority data and a randomly sampled subset of the majority data in $\mathcal{D}^{(t)}$. The algorithm can work with a variety of supervised algorithms as its base classifier. In this effort, we use the multi-layer perceptron (MLP). Note however, unlike Learn^{++}.NSE, Learn^{++}.NIE creates a sub-ensemble of classifiers for each new batch of data with two primary reasons: i) ensembles can reduce error through the averaging obtained through the weighted voting, and more importantly, ii) choosing a random subset of majority class for training each classifier provides a less imbalanced dataset for training the classifier, while using an ensemble of such classifiers allows us to minimize the loss of information that may be caused by using a subset of the data for training.

Algorithm: Learn^{++}.NIE

Input: Training data $\mathcal{D}^{(t)} = \left\{\boldsymbol{x}_i^{(t)} \in \mathbf{X}, y_i^{(t)} \in \Omega\right\}$, $\Omega = \{+1, -1\}$ $i = 1, \ldots, m^{(t)}$; Supervised learning algorithm, *BaseClassifier*; Number of classifiers in sub-ensemble, K (3); Error weight η (0.5), $0 \leq \eta \leq 1$; Sigmoid parameters, a (0.5) and b (15), $a, b \in \mathbb{R}$;

for $t = 1,2, \ldots$ (as long as new datasets arrive)

1. Call $h_t = BaggingVariation(base\ classifier, \mathcal{D}^{(t)}, K)$
2. Evaluate all exiting sub-ensembles on new dataset, $\mathcal{D}^{(t)}$, where $k = 1,2, \ldots, t$ (t is the most recent / current time step)
   Call $E_k^{(t)} = MajorityVote\left(h_t, \mathcal{D}^{(t)}\right)$ and compute class recalls from $E_k^{(t)}$ where $k = 1,2,3, \ldots, t$
   $$r_{k,(+)}^{(t)} = \frac{tp}{tp + fn} \qquad r_{k,(-)}^{(t)} = \frac{tn}{tn + fp}$$
   $$\epsilon_k^{(t)} = \eta\left(1 - r_{k,(+)}^{(t)}\right) + (1-\eta)\left(1 - r_{k,(-)}^{(t)}\right)$$
   if $\epsilon_k^{(t)} > 1/2$ **then** $\epsilon_k^{(t)} = 1/2$ **end if**
   $$\beta_k^{(t)} = \epsilon_k^{(t)} / \left(1 - \epsilon_k^{(t)}\right)$$
3. Compute a weighted sum of all weighted error for each sub-ensemble where $k = 1,2, \ldots, t$
   $$\omega_k^{(t)} = 1/(1 + e^{-a(t-k-b)})$$
   $$\omega_k^{(t)} = \omega_k^{(t)} / \textstyle\sum_{j=0}^{t-k} \omega_k^{(t-j)},$$
   $$\hat{\beta}_k^{(t)} = \sum_{j=0}^{t-k} \omega_k^{(t-j)} \beta_k^{(t-j)}$$
4. Calculate classifier voting weights
   $$W_k^{(t)} = \log\left(1/\hat{\beta}_k^{(t)}\right)$$
5. Obtain the composite hypothesis
   $$H^{(t)}(\boldsymbol{x}_i) = \arg\max_c \textstyle\sum_k W_k^{(t)} \left[\!\left[E_k^{(t)}(\boldsymbol{x}_i) = c\right]\!\right]$$

endfor

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

All existing sub-ensembles are then evaluated on the most recent data, $\mathcal{D}^{(t)}$. Note that each $h_{k=1,\ldots,t}$ contains K classifiers that are combined using majority voting. The predicted class labels are $E_k^{(t)}$, on the data at the most recent time step ($\mathcal{D}^{(t)}$) on which the kth sub-ensemble (h_k) is being evaluated. $E_k^{(t)}$ is a vector containing all predicted class labels for the training data in $\mathcal{D}^{(t)}$ by the kth sub-ensemble. The recall of each class (majority & minority) is computed in step (2) from $E_k^{(t)}$, where tp, tn are true positive and true negative ratios, and fp, fn are the false positive and false negative ratios. The two recall measures are then combined using the weighted average controlled by η in Eq. 1 (step 2). If the weighted error, $\epsilon_k^{(t)}$, exceeds ½, it is set to ½ which yields a normalized error of 1, and an associated voting weight of 0 (step 4).

TABLE I. MEAN AND STANDARD DEVIATION GAUSSIAN DRIFT OVER TIME

	$t = 0\ to\ t = 1/3$				$t = 1/3\ to\ t = 2/3$				$t = 2/3\ to\ t = 1$			
	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y	σ_x	σ_y	μ_x	μ_y
$C_{1,1}$	1	$1+6t$	2	5	1	3	2	5	1	$8-9(t-1/3)$	$8-9(t-1/3)$	$8-9(t-1/3)$
$C_{1,2}$	$3-6t$	1	5	8	1	1	$5+9(t-1/3)$	8	1	1	8	8
$C_{1,3}$	$3-6t$	1	5	2	1	1	$5+9(t-1/3)$	2	1	1	8	2
$C_{2,1}$	1	1	8	5	1	1	$8-9(t-1/3)$	5	1	1	$8-9(t-1/3)$	$8-9(t-1/3)$

The original Learn^{++}.NSE algorithms computes a pseudo error by providing a higher penalty to those classifiers that misclassify the instances that are misclassified by the current ensemble. This means that the misclassification of certain instances, namely, those that have been misclassified by the old ensemble, are costlier than others. Learn^{++}.NIE, however, does not reduce a classifier's weight for misclassifying a particular instance; rather it uses the overall error of each class on the most recent data. Prior to computing the weights for each sub-ensemble decision, the normalized error is weighted using a logistic sigmoid function giving more weight to the most recent recall measures (step 3). The voting weight for each sub-ensemble is then the logarithm of the age-adjusted weighted error average (step 4). The ensemble decision is obtained using a weighted majority vote in step 5 to obtain the final hypothesis, $H^{(t)}(\boldsymbol{x}_i)$.

IV. EXPERIMENTAL RESULTS

We provide a comparison of Learn^{++}.NIE, Learn^{++}.NSE, and Learn^{++}.SMOTE (a combination of SMOTE and Learn^{++}.NSE) to determine the advantages and disadvantages of each algorithm on a variety of nonstationary environments. In our implementation of SMOTE, the number of nearest neighbors was set to 9 and the amount of SMOTE was set to 300 for the SEA data and 1500 for the Gaussian data. All algorithms are compared to each other overall performance, f-measure and recall of the minority class.

A. Gaussian Data

In order to precisely control the nonstationary environment, as well as to be able to compare results to that of a Bayes classifier, we created a drifting Gaussian dataset with approximately 3% minority data. The majority class was designed as a multimodal (linear combination of 3 modes) distribution, whereas the minority class came from a unimodal distribution, as shown in Figure 2 (the z-axis represents the likelihoods of the data). The drift was introduced by varying the mean and variance of each class distribution with time, according to the parametric equations given in Table I, where $C_{c,m}$ denotes class (c) and mode (m), with $c = 2$ representing the minority class.

The Bayes decision region can be seen in Figure 3 where the z-axis represents the posterior probability. The light gray (cyan) colored areas of the feature space represent the posterior probability of the minority class, which moves through the middle of the 3-modes of the majority class (dark gray / purple shaded regions) throughout the experiment.

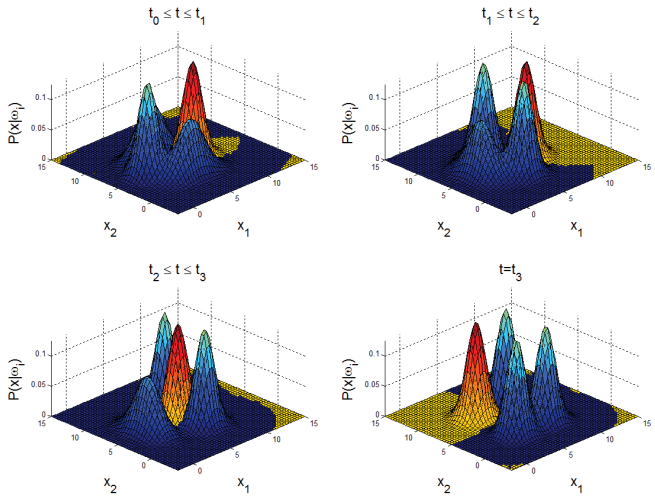

Fig. 2. Likelihoods, $\boldsymbol{P(x|\omega_i)}$, used to compute the Bayes classifier of the Gaussian data presented in TABLE I. (a) Initial distribution at $\boldsymbol{t = t_0}$, (b) some later time step $\boldsymbol{t_1 > t_0}$, (c) time step $\boldsymbol{t_2 > t_1}$ and (d) end point at $\boldsymbol{t = t_3}$.

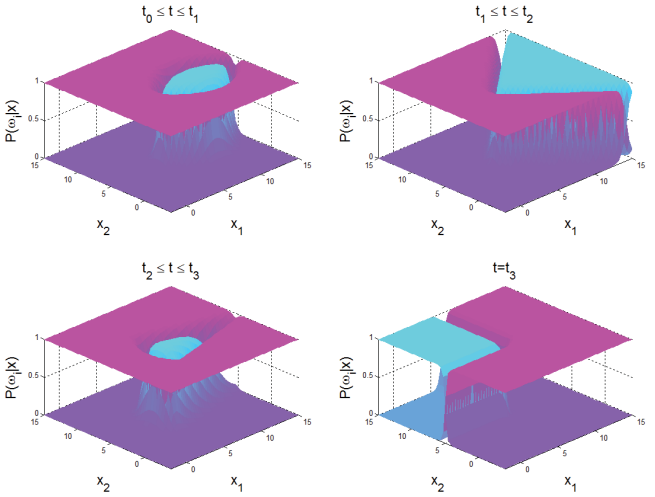

Fig.3. Posterior probability, $\boldsymbol{P(\omega_i|x)}$, computed by the Bayes classifier where the pink (dark) region is the majority class and the cyan (light) region is the minority class for the Gaussian data in TABLE I. (a) Posterior at $\boldsymbol{t = t_0}$, (b) some later time step $\boldsymbol{t = t_1 > t_0}$, (c) time step $\boldsymbol{t_2 > t_1}$ and (d) end point at $\boldsymbol{t = t_3}$.

Each sub-ensemble generated three multi-layer perceptrons (20 hidden layer nodes with sigmoid activation functions); with error weighting sigmoid parameters of a=0.5 and b =15. Each batch of training/testing data contained 1500 majority and 50 minority examples. The performance, f-measure, and recall of the three different algorithms – along with that of Bayes classifier – are shown in Figure 4, 5, and

6 respectively. The shading around each curve indicate the 95% confidence interval (α=0.05) based on 25 independent trials. The η term was set to 0.5 making the penalty for both minority and majority class recall error the same. We later show a more detailed empirical analysis of the variation of η. Overall, the performances of all algorithms were comparable to that of each other (as expected, Bayes classifier performing best, followed by the original Learn^{++}.NSE) with little or no statistically significant differences – except at $t = 55{\sim}65$. This drop in performance occurs when the minority class is surrounded by the modes of the majority class, thus making the minority class prediction the most difficult.

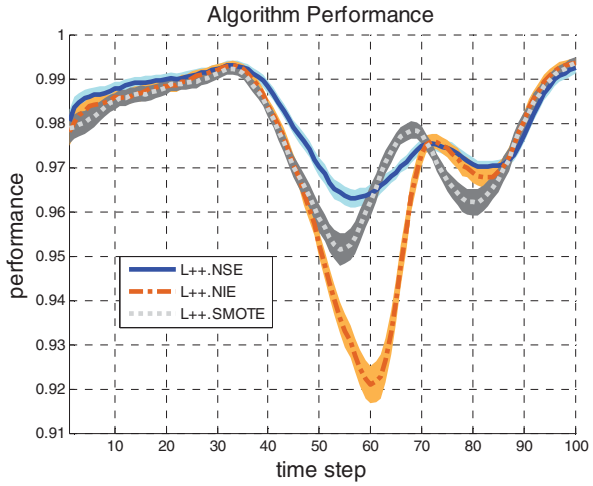

Fig. 4. Classification performance comparison on Gaussian data.

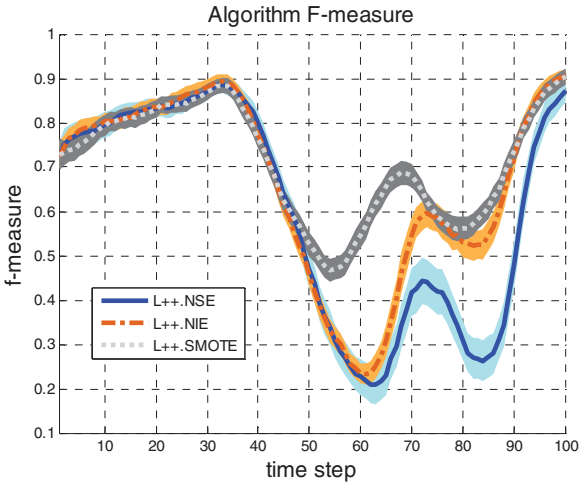

Fig. 5. f-measure comparison on Gaussian data.

The real benefit of Learn^{++}.NIE, or of adding SMOTE to the Learn^{++}.NSE framework, can be seen in f-measure and recall characteristics. Learn^{++}.SMOTE has the best minority class recall as well as best f-measure, perhaps because SMOTE can directly modify the feature space by creating more synthetic minority examples to learn from, which is particularly effective when the minority class is located in the center of the majority class (Figure 2.c).

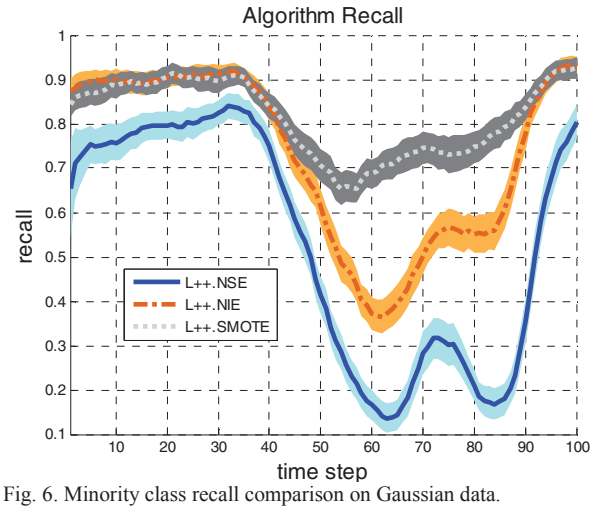

Fig. 6. Minority class recall comparison on Gaussian data.

Learn^{++}.NIE does exhibit significantly better recall and f-measure of the minority class than Learn^{++}.NSE, though it cannot match Learn^{++}.SMOTE on this dataset for the default value of $\eta = 0.5$. However, the η term in Learn^{++}.NIE does provide a meaningful control on the algorithm behavior. The effect of varying the η term in the Learn^{++}.NIE algorithm can be viewed in the recall and f-measure characteristics in Figure 7 and 8, respectively.

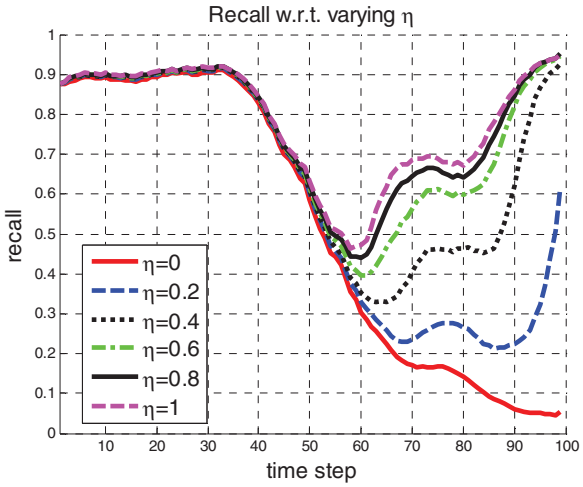

Fig. 7. The effect of varying $\boldsymbol{\eta}$ on recall of Learn^{++}.NIE on the Gaussian dataset.

Note that η=0 assigns no penalty to misclassifying a minority class instance and the only contribution to weighted error come from a mistake made on the majority class. Therefore, the recall of the minority class is worst when η=0. The extremely low recall also results in a low f-measure. Increasing η yields a larger recall of the minority class, as well as f-measure. In fact, for $\eta \geq 0.6$, Learn^{++}.NIE catches up with Learn^{++}.SMOTE. Of course, one should always be aware of the trade-off between recall and overall classification performance, as better minority recall is typically associated with poorer classification performance on the majority class.

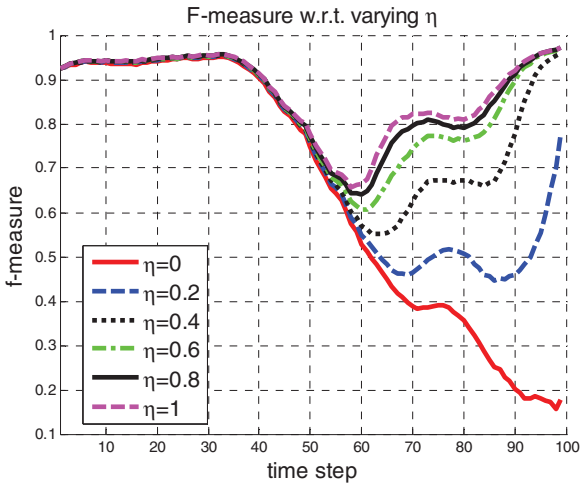

Fig. 8. The effect of varying η on the f-measure of Learn^{++}.NIE on the Gaussian dataset.

In summary, we conclude that we can choose a value of η that provides a good balance between performance, precision, recall, and f-measure. Learn^{++}.NIE does outperform Learn^{++}.NSE in recall and f-measure, but Learn^{++}.SMOTE provides the best recall. We believe this is due to SMOTE's ability to appropriately modify the feature space through its pseudo oversampling. Figure 3c shows the interval when the minority decision space is the smallest, which is also the interval when we observe the drop in recall/f-measure for Learn^{++}.NSE/NIE, and a boost in recall with SMOTE.

B. *SEA Data*

We also evaluate the algorithms on the commonly used benchmark SEA dataset introduced by Street and Kim [10]. SEA dataset uses 3-dimenisional data, only two of which carry information, and a shifting hyperplane with 5% class noise added to the training and testing datasets (in addition to the third feature being noise). We modify this dataset to make it imbalanced, and to introduce a cyclical drift, where the hyperplane shifts back and forth between two different thresholds for two cycles. This shifting hyperplane also causes the class imbalance to vary between 7% and 25% as the hyperplane shifts. The performance, f-measure and recall plots for each algorithm are shown in Figure 9, 10, and 11 respectively. The base classifier used in this experiment was a decision tree. Three classifiers were generated in each sub-ensemble. The original results of SEA algorithm on this dataset, as well as those of original Learn^{++}.NSE using other base classifiers can be found in [10;16;22].

We make several observations. First, Learn^{++}.NSE and Learn^{++}.SMOTE only have a small change in performance when the hyperplane shifts for the second time. Learn^{++}.NIE, on the other hand, experiences a much larger drop in performance. Second, Learn^{++}.NIE takes longer to recover after the concept change, though, the recovery in performance and the classification accuracy is significantly higher than Learn^{++}.NSE. Third, Learn^{++}.SMOTE has promising performance results with the proper selection of the SMOTE parameters, but the boost in recall was not as significant as Learn^{++}.NIE. The Learn^{++}.SMOTE recall can of course be increased by increasing the percentage of SMOTE, however, the algorithm will then generally begin to experience a degradation in performance. After all, as the amount of SMOTE increases, the imbalance reduces, and at the extreme case, the minority class becomes majority by oversampling too many minority class examples.

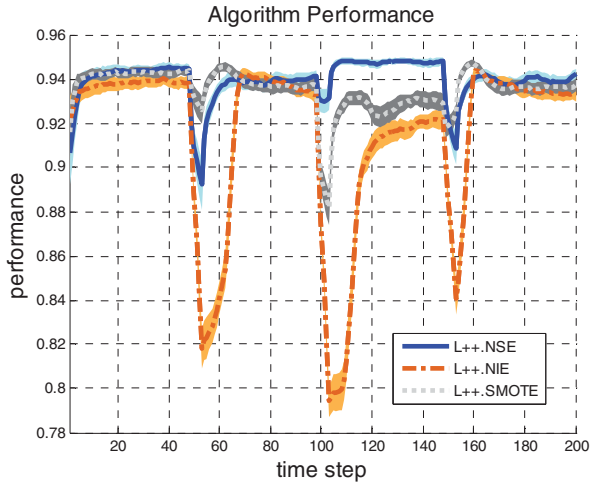

Fig. 9. Algorithm performance on the SEA dataset with a 95% confidence interval.

The Learn^{++}.SMOTE combination still maintains a better recall performance compared to the original Learn^{++}.NSE, similar to the Gaussian data experiment (figure 10). However, Learn^{++}.NIE generally outperforms both other algorithms in f-measure, and in certain time steps of recall, although there is a significant drop caused by the concept change (albeit with a slower recovery rate). Our conclusion from the SEA dataset is that the Learn^{++}.NIE generally outperforms in recall and f-measure (and is competitive for performance with) both its predecessor Learn^{++}.NSE (expected, since Learn^{++}.NSE is not designed to handle imbalanced data), and the Learn^{++}.SMOTE combination (somewhat pleasantly surprising, since SMOTE and Learn^{++}.NSE each can handle their respective imbalance data and concept drift tracking extremely well). On the other hand, Learn^{++}.NIE appears to have a weak point when there is sudden concept change, especially compared to Learn^{++}.NSE, which was shown to recover quickly from the concept change with different base classifiers [22]. Not only is this observed in the performance of the algorithm but also in the recall.

The effect of varying η in Learn^{++}.NIE can be seen in recall and f-measure in Figure 12 and 13, respectively. When $\eta = 0$ only the majority class error contributes to determining the weight of the sub-ensemble and when $\eta = 1$, the minority class error is the only contributor to the sub-ensemble weight. Varying this term, we observe that η can be used to control the recall of the minority class and the overall classification performance of the algorithm. If there is prior knowledge available, an appropriate η value can be determined to find a balance between the recall and overall performance of the algorithm.

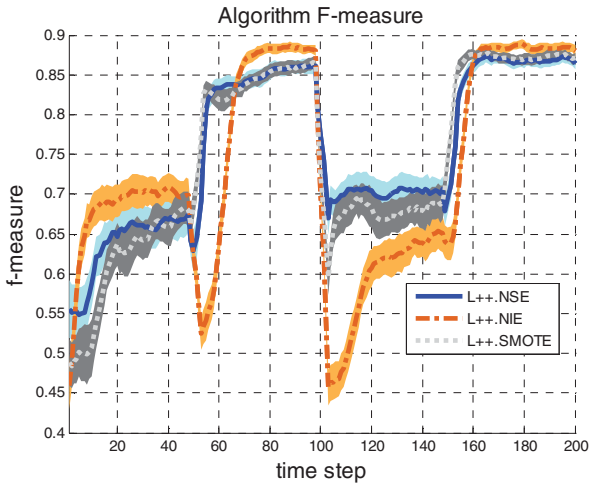

Fig. 10. f-measure comparison on the SEA dataset.

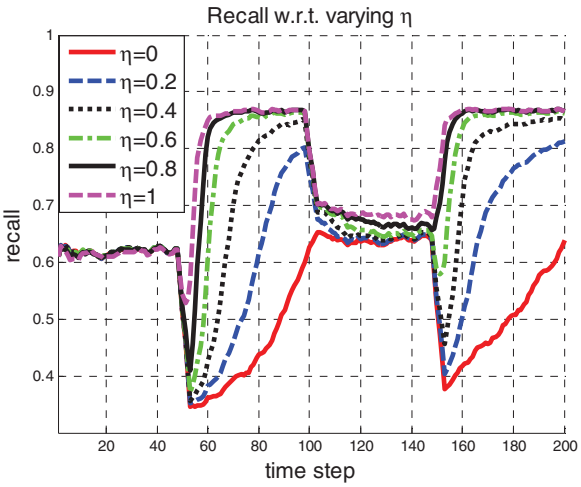

Fig. 12. The effect of η on recall of Learn^{++}.NIE on SEA dataset.

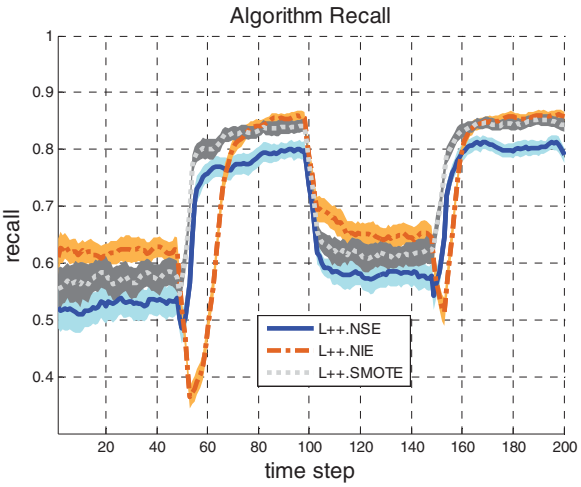

Fig. 11. Recall comparison on the SEA dataset.

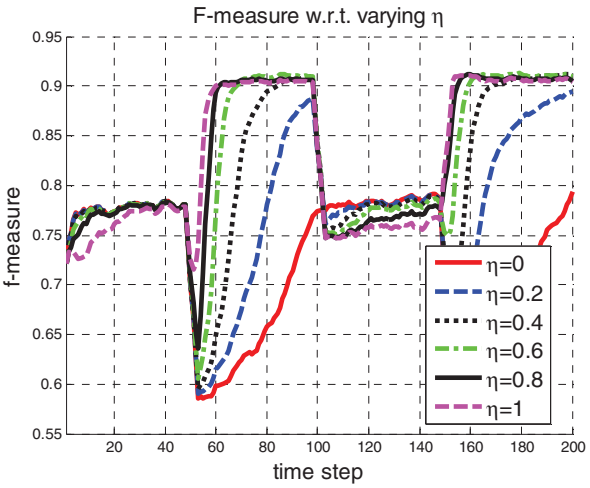

Fig. 13. The effect of η on f-measure of Learn^{++}.NIE on SEA dataset.

We also conclude on this dataset that the Learn^{++}.NIE generally recalls minority class data better than the Learn^{++}.NSE. Adding SMOTE to Learn^{++}.NSE, however, yields a statistically significant boost in recall; but the result is not nearly as significant as Learn^{++}.NIE. On the other hand, we should also add that unlike SMOTE (or other imbalanced data approaches), Learn^{++}.NIE does not generate any extra minority points – synthetic or otherwise – and utilizes only the instances available in the dataset.

V. Conclusion

We have introduced two new members of the Learn^{++} family of incremental learning algorithms, Learn^{++}.NIE, and Learn^{++}.SMOTE designed to work with data experiencing concept drift and class imbalance at the same time. As incremental learning algorithms, neither Learn^{++}.NIE, nor Learn^{++}.SMOTE requires access to any of the previous data, unlike other algorithms developed for similar goals [11;12].

Learn^{++}.NIE is more favorable at boosting minority class performance than Learn^{++}.NSE and is comparable to (or sometimes better, based on the η value than) Learn^{++}.SMOTE. This was demonstrated on two controlled experiments. It is worth noting that using an algorithm of this nature, where overall error is no longer the primary metric in determining the weight of a classifier, the performance of the algorithm is not expected to be superior compared to algorithms like Learn^{++}.NSE, and this was observed in both experiments. Such superiority may or may not be meaningful depending on the dataset. However, the advantage of this new approach is that the performance on the minority class – usually the class of particular importance – increases significantly when the data is severely imbalanced. The weak point of Learn^{++}.NIE is its relatively slow recovery – compared to the original Learn^{++}.NSE, from sudden concept change, as seen on the SEA dataset. Learn^{++}.NSE handles this change well because a lower voting weight is assigned to classifiers that misclassify examples that the previous ensemble had misclassified. This method of weighting allows for Learn^{++}.NSE to quickly remove classifiers that are unable to predict on the new concept by assigning a lower weight to them and thus leading to a quick recovery in performance. In an imbalanced data scenario, however, such an error measure is not suitable as a reliable figure or merit.

The Learn^{++}.NIE algorithm was generally more capable of recalling the minority class with statistical significance, compared to the original Learn^{++}.NSE algorithm, as observed on both datasets. The proposed framework was able to recall significantly more of the SEA minority class. In addition, the Learn^{++}.NIE algorithm also had favorable results over Learn^{++}.NSE integrated with SMOTE on the noisy SEA dataset. We have also shown that the simple combination of Learn^{++}.NSE and SMOTE works well for datasets that experience concept drift and class imbalance.

Perhaps the most important contribution of Learn^{++}.NIE is that it allows control over how much penalty is given to the error of the majority and minority class recall separately, through a weighted average error. The algorithm can reward classifiers that are performing well on both minority and majority classes rather than just the majority class. The η term effectively allows choosing a balance between recall of the minority class and overall performance of the algorithm. In the absence of prior knowledge, this parameter can easily be set to the default value of 0.5 for a good balance between the recall and overall performance of the algorithm.

Future work will include an analysis of the Learn^{++}.NIE algorithm with different statistical measures like f-measure or g-mean to weight the sub-ensembles. Other base classifiers will need to be evaluated with this algorithm. More datasets, both synthetic and real-world, will need to be evaluated with this algorithm to determine the strengths and weakness of the proposed approach.

REFERENCES

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

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and M. A. Khasawneh, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, June2002.

[3] M. Muhlbaier and R. Polikar, "An Ensemble Approach for Incremental Learning in Nonstationary Environments," *7th. Int. Workshop on Multiple Classifier Systems (MCS2007)* in Lecture Notes in Computer Science, vol. 4472, Berlin: Springer, pp. 490-500, 2007.

[4] G. Jing, B. Ding, F. Wei, H. Jiawei, and P. S. Yu, "Classifying Data Streams with Skewed Class Distributions and Concept Drifts," *Internet Computing, IEEE*, vol. 12, no. 6, pp. 37-49, 2008.

[5] C.Alippi, B. G, and M.Roveri, "Just in time classifiers: managing the slow drift case,", pp. 114-120, 2009.

[6] 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.

[7] L.I.Kuncheva, "Using Control Charts for Detecting Concept Change in Streaming Data," School of Computer Science, Bangor University, UK,BCS-TR-001-2009, 2009.

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

[9] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: Overview and perspectives," *European Conference on Artificial Intelligence (ECAI)*, pp. 5-10, 2008.

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

[11] J. Gao, W. Fan, J. Han, and P. S. Yu, "A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions," *SIAM International Conference on Data Mining*, vol. 7, 2007.

[12] S.Chen and H.He, "SERA: Selectively Recursive Approach towards Nonstationary Imbalanced Stream Data Mining," *International Joint Conference on Neural Networks*, Atlanta, GA: pp. 522-529, 2009.

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

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

[15] M. D. Muhlbaier and R. Polikar, "Multiple Classifiers Based Incremental Learning Algorithm for Learning in Nonstationary Environments," *IEEE International Conference on Machine Learning and Cybernetics (ICMLC 2007)*, vol. 6, pp. 3618-3623, 2007.

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

[17] M.Kuba, R.Holte, and S.Matwin, "Machine Learning for the Detection of Oil Spills in Satellite Radar Images ," *Machine Learning*, vol. 30, pp. 195-215, 1998.

[18] Haibo He and Edwardo Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2010.

[19] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, " SMOTEBoost: Improving Prediction of the Minority Class in Boosting," *7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pp. 107-119, 2003.

[20] C.Li, "Classifying imbalanced data using a bagging ensemble variation (BEV)," *Proceedings of the 45th annual ACM Southeast Regional Conference*, 2007.

[21] R. Elwell and R. Polikar, "Incremental Learning of Variable Rate Concept Drift," *8th International Workshop on Multiple Classifier Systems (MCS 2009)*in Lecture Notes in Computer Science, eds. J. A. Benediktsson, J. Kittler, and F. Roli, Eds., vol. 5519, pp. 142-151, 2009.

[22] M. Karnick, M. D. Mulhbaier, and R. Polikar, "Incremental Learning in Non-Stationary Environments with Concept Drift Using a Multiple Classifier Based Approach," *International Conference on Pattern Recognition (ICPR 2008)*, pp. 1-4, 2008.