2010 International Conference on Pattern Recognition

An Incremental Learning Algorithm for Non-Stationary Environments and Class Imbalance

Gregory Ditzler, Robi Polikar
Electrical & Computer Engineering,
Rowan University
Glassboro, NJ, 08028 USA
ditzle53@students.rowan.edu, polikar@rowan.edu

Nitesh Chawla
Computer Science & Engineering,
Notre Dame University
South Bend, IN, 46556 USA
nchawla@cse.nd.edu

Abstract – **Learning in a non-stationary environment and in the presence of class imbalance has been receiving more recognition from the computational intelligence community, but little work has been done to create an algorithm or a framework that can handle both issues simultaneously. We have recently introduced a new member to the Learn^{++} family of algorithms, Learn^{++}.NSE, which is designed to track non-stationary environments. However, this algorithm does not work well when there is class imbalance as it has not been designed to handle this problem. On the other hand, SMOTE – a popular algorithm that can handle class imbalance – is not designed to learn in nonstationary environments because it is a method of oversampling the data. In this work we describe and present preliminary results for integrating SMOTE and Learn^{++}.NSE to create an algorithm that is robust to learning in a non-stationary environment and under class imbalance.**

Keywords-Multiple Classifier Systems, nonstationary environments, class imbalance

I. INTRODUCTION

Learning in non-stationary environments (NSE), also called concept drift, involves learning from streaming data with a changing underlying distribution. Most existing algorithms (most neural networks, SVMs, decision trees, Adaboost, etc.) are simply not equipped to handle drifting data streams. Several existing approaches for learning concept drift use a sliding time window for filtering incoming data, and updating an existing classifier on the most recent data segment, a method introduced by FLORA family of algorithms [1]. In these approaches, the old data – and the classifiers trained on such data - are simply assumed irrelevant and are discarded. Another group of approaches employ ensemble of classifiers. For example, Kolter and Maloof's dynamic weighted majority (DWM) uses online naïve Bayes classifiers trained incrementally to classify in NSE by adding new dynamically weighted classifiers and removing those whose weight drops below a threshold [2]. Gao presents a general framework based on bagging and using previous data to train classifiers [3], Nishida suggests adjusting the structure of the ensemble in ACE [4], and Street proposed the streaming ensemble algorithm (SEA) to replace the classifiers in an ensemble that are contributing the least [5]. The Learn^{++}.NSE, briefly described later in this paper, also uses an ensemble of classifiers, but has the unique property of recalling old classifiers when a cyclical environment makes previous classifiers relevant [6]. This algorithm was shown to work well in a variety of problems and different drift rates [7].

However, only the bagging framework presented in [3] is designed to also handle the class imbalance problem, but it cannot learn the environment without access to previous data. The primary challenge in learning from imbalanced data is to accurately predict minority class instances without defaulting to the majority. Basic methods to work around class imbalance include undersampling and oversampling the majority and minority classes, respectively. However, undersampling is unattractive because it throws data away and oversampling leaves a classifier prone to overfitting the minority class. SMOTE presented by Chawla in [8], takes a novel approach to resampling: rather than modifying the data space, SMOTE modifies the feature space by creating synthetic examples that lie on the line segment between the neighboring minority examples. SMOTE has been shown to handle severe data imbalance in a variety of synthetic and real world scenarios.

In this contribution, we evaluate the feasibility of what appears to be an obvious solution to handle the combined problem of learning concept drift from imbalanced data: combining SMOTE and Learn^{++}.NSE. The rest of the paper is organized as follows. Section II will highlight the Learn^{++}.NSE algorithm modified for imbalanced datasets. Section III will describe the datasets used and the results. Finally, Section IV brings together conclusions from the experiments.

II. LEARN^{++}.SMOTE

Learn^{++}.SMOTE, shown in Fig. 1, integrates the incremental concept drift learning algorithm Learn^{++}.NSE with SMOTE. Since this is an incremental learning algorithm, we assume that we do not have access to the previous data (minority class included). The primary free parameters of this algorithm are the base classifier, the sigmoid weighting parameters, the amount of SMOTE (%) to add into the dataset and the number of nearest neighbors of minority samples used for creating synthetic samples. The base classifier can be any supervised learning algorithm.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.734

The algorithm is provided data incrementally: at time t, dataset $\mathfrak{D}^{(t)}$, comes from a distribution $p^{(t)}(\boldsymbol{x}, y_k)$ which may be different than $p^{(t-1)}(\boldsymbol{x}, y_k)$. Learn++.SMOTE then updates a distribution of instance weights by evaluating the existing ensemble hypothesis, $H^{(t-1)}$, on the most recent dataset, $\mathfrak{D}^{(t)}$.The weights of those instances misclassified are increased and renormalized to create the instance weighting distribution $D^{(t)}$. A new classifier is trained on $\mathfrak{D}^{(t)}$ and a new synthetic data subset is created by calling the SMOTE algorithm. The new classifier, h_t, and all previously generated classifiers in the ensemble are evaluated on this new dataset to obtain their errors on the new environment. If the error for the new classifier is greater than ½, it is discarded and a new one is created; however, if an older classifier's error is greater than ½ its error is set to ½, which results in a zero voting weight after the weight normalization. The reason for this double standard is the idea that an older classifier may be currently underperforming but may become relevant again later if the environment follows a cyclical nature.

A logistic sigmoid is then applied to errors of each classifier across all time steps. Parameter a defines the slope of the sigmoid cutoff, and b refers to the number of prior errors to be considered before the cutoff. This style of weighting will reward classifiers that are currently performing well on the most recent environments, even if such classifiers may have been generated long time ago. Therefore, if a classifier had its error previously set to ½, or the error is ½ at the current time step, its ensemble voting weight may not be set to zero because the previous errors of the classifier also contribute to the final voting weight, with the most recent errors given the highest weight. The final ensemble decision is obtained using weighted majority voting of all classifiers whose weights are normalized based on their average performance, biased to more recent environments. Ensemble pruning can be added into the algorithm to limit the size of the final ensemble, but has been omitted for brevity in this effort. See reference [9] for the effects of pruning with Learn^{++}.NSE.

Based on our previous experience, the a and b parameters of the logistic sigmoid in the Learn^{++}.NSE algorithm were chosen as 0.5 and 15 for all test presented in this work, respectively.

Algorithm: Learn^{++}.SMOTE

Input: Training data $\mathfrak{D}^{(t)} = \left\{\boldsymbol{x}_i^{(t)} \in \mathbf{X}, y_k^{(t)} \in \Omega\right\}$, $i = 1,2,\dots,m^{(t)},\ k = 1,2,\dots,C$
Supervised learning algorithm, BaseClassifier
for $t = 1,2,\dots$
if $t = 1,\ D^{(1)} = w^{(1)}(i) = 1/m^{(t)},\ \forall i$
endif

1. Compute current ensemble error
$$E^{(t)} = \sum_{i=1}^{m^{(t)}} \left(1/m^{(t)}\right) \cdot [\![H^{(t-1)}(\boldsymbol{x}_i) \neq y_i]\!]$$
2. Updated and normalize instance weights
$$w_i^{(t)} = \frac{1}{m^{(t)}} \cdot \begin{cases} E^{(t)} & H^{(t-1)}(\boldsymbol{x}_i) = y_i \\ 1 & otherwise \end{cases}$$
$$D^{(t)} = w^{(t)} \Big/ \sum_{i=1}^{m^{(t)}} w_i^{(t)}$$
3. Call SMOTE on minority class to create $\mathcal{S}^{(t)}$
4. Call BaseClassifier with $\mathfrak{D}^{(t)}$ and $\mathcal{S}^{(t)}$, obtain $h_t: \mathbf{X} \to Y$
5. Evaluate all exiting classifiers on new dataset $\mathfrak{D}^{(t)}$
$$\epsilon_k^{(t)} = \sum_{i=1}^{m^{(t)}} D^{(t)}(i)[\![h_k(\mathbf{x}_i) \neq y_i]\!] \quad \text{for } k = 1,\dots,t$$
If $\epsilon_k^{(t)} > 1/2$, geneate a new h_t. If $\epsilon_{k<t}^{(t)} > 1/2$, set
$$\epsilon_{k<t}^{(t)} = 1/2 \quad \beta_k^{(t)} = \epsilon_k^{(t)} \Big/ \left(1 - \epsilon_k^{(t)}\right)$$
6. Compute a weighted sum of all normalized error for the kth classifier h_k
$$\omega_k^{(t)} = 1/(1 + e^{-a(t-k-b)})$$
$$\omega_k^{(t)} = \omega_k^{(t)} \Big/ \sum_{j=0}^{t-k} \omega_k^{(t-j)},\ \hat{\beta}_k^{(t)} = \sum_{j=0}^{t-k} \omega_k^{(t-j)} \beta_k^{(t-j)}$$
7. Calculate classifier voting weights
$$W_k^{(t)} = \log\left(1/\hat{\beta}_k^{(t)}\right)$$
8. Obtain the composite hypothesis
$$H^{(t)}(\boldsymbol{x}_i) = \arg\max_c \sum_k W_k^{(t)} [\![h_k(\boldsymbol{x}_i) = c]\!]$$

Figure 1. Learn^{++}.NSE-SMOTE algorithm

The SMOTE parameters, number of nearest neighbors (k) and percentage of oversampling (N) for the Gaussian dataset were chosen as $k = 9$ and $N = 1500$, whereas for the electricity pricing dataset (*elec2*) these parameters were $k = 9$ and $N = 1500$. The datasets are briefly described in the next section. All results include a 95% confidence interval around each of the measures used to assess the quality of the algorithms tested for nonstationary learning and class imbalance.

TABLE I. PARAMETRIC EQUATIONS FOR GAUSSIAN DRIFT

	$t = 0\ to\ t = 1/3$				$t = 1/3\ to\ t = 2/3$				$t = 2/3\ to\ t = 1$			
	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	$8-9(t-1/3)$	$8-9(t-1/3)$

III. Implementation & Results

The proposed algorithm was tested on two different databases, one synthetic and one real-world, that are known to be nonstationary and that involve class imbalance. These datasets are the synthetic Gaussian data whose drift is adjusted by the parametric equations in Table I, and the real-world electricity pricing dataset (*elec2*) obtained from [10]. The synthetic Gaussian dataset consists of a class imbalance such that the data is ≈3% minority. The class imbalance ratio remains constant for the duration of this test. The majority class is made up of three modes and the minority is a single mode distribution. The primary benefit of this dataset is that it allows us to set and control the drift, and also allows us to compute Bayes error for comparison. The mean and standard deviations of the classes can be found in Table 1. Note that $C_{c,m}$ is a notion of the class (c) and mode (m) where $c = 2$ is the minority.

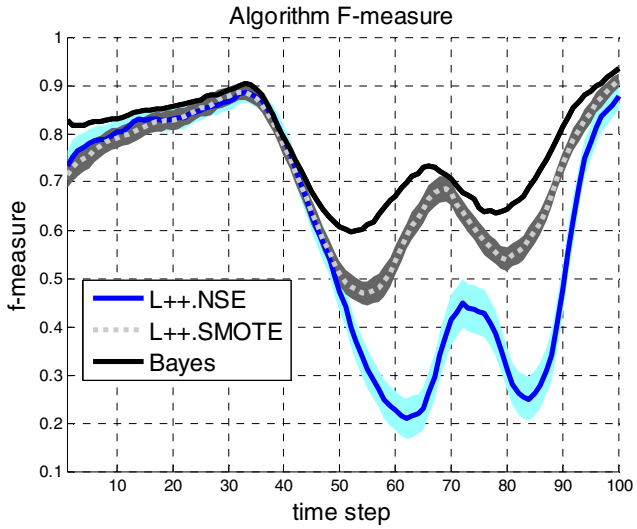

Figure 2. F-measure on synthetic data

The f-measure and recall are used as evaluation metrics for the algorithms on each dataset. We note that simple accuracy is not a reliable indicator in such severely imbalanced datasets, as blindly predicting the majority class will always lead to (misleading) high performance due to the class imbalance. In such imbalanced dataset, the performance on the minority class (recall and precision) are often more important. The f-measure metric is a balanced combination of precision and recall of the minority class. A MLP was used as the base classifier with a 20x2 architecture and sigmoid activation functions. Fig. 2 and 3 compare the Bayes classifier performance, using a 1-0 loss function, to the original Learn^{++}.NSE and Learn^{++}.SMOTE using the f-measure and recall metrics on the synthetic data, respectively. The shaded regions around performance curves indicate the 95% confidence intervals across 25 independent trials. Also note that results of SMOTE alone are not included as SMOTE – with any base classifier –will not be able to learn the drifting environment incrementally.

As expected, the f-measure for the Bayes classifier is consistently best across all times, though Learn^{++}. SMOTE follows Bayes performance very closely with little or no statistical significance both in f-measure and recall. The original Learn^{++}.NSE has a significantly poorer f-measure and recall in most time steps compared to both the Bayes and Learn^{++}.SMOTE algorithms. These results demonstrate the effectiveness of adding SMOTE to Learn^{++}.NSE in class imbalance cases.

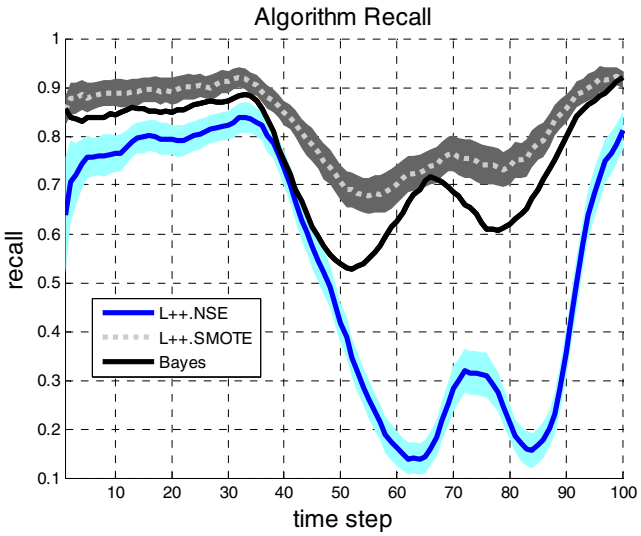

Figure 3. Recall on synthetic data

Our second experiment used the *elec2* dataset, which contains inherent drift. All examples with missing features have been removed from the dataset during preprocessing. However, there is relatively little class imbalance in the data, so the *minority* class was generated by undersampling one of the classes to bring the imbalance to ≈1:17 (making the problem much more difficult than it originally is). Note that the class imbalance ratio will vary from training set to training set since this is not a controlled experiment. The datasets are presented to the algorithm in chunks of 225 examples for training; the next 225 examples are used for evaluation of the learning algorithm at each time step. A MLP was used as the base classifier with 55 hidden layers nodes using sigmoid activation functions. Fig. 4 shows the f-measure of both Learn^{++}.NSE and Learn^{++}.SMOTE. The primary observation to make with this plot is that neither algorithm appears to have a dominantly better f-measure.

However, adding SMOTE to the original Learn^{++}. NSE algorithm (Fig. 5) added a statistically significant gain in recall on the minority class, especially in later time steps. While there is an increase in the recall when SMOTE was added, the precision generally observed a slight drop (not shown due to space limitations, but the drop was generally not significant at every time step) compared to the original NSE algorithm which is generally expected.

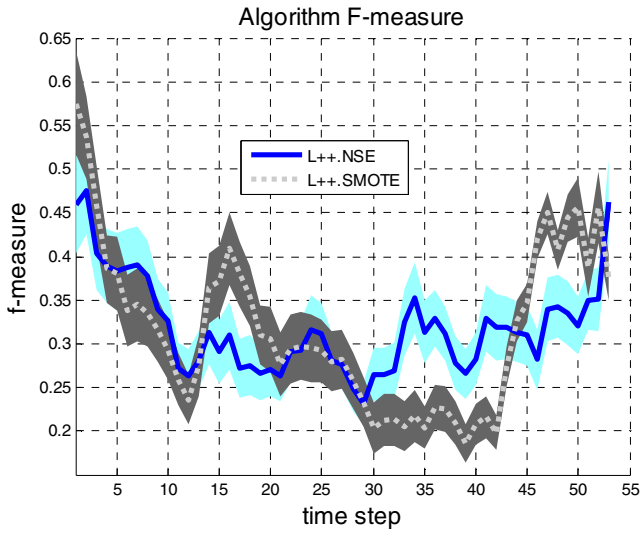

Figure 4. F-measure on synthetic data

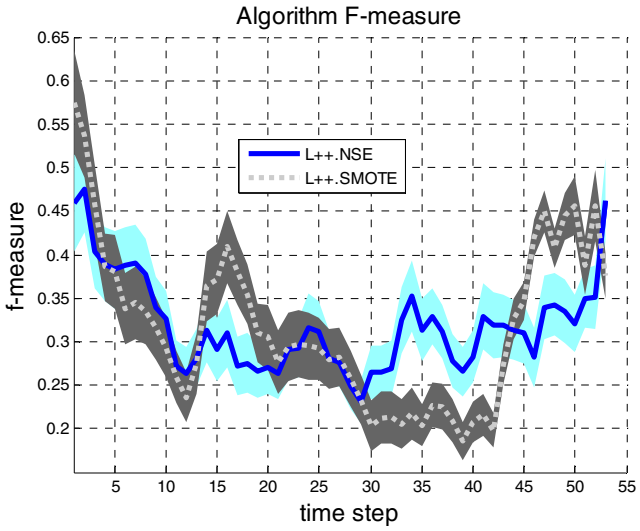

Figure 5. Recall-score on synthetic data

IV. Conclusions

We have presented a hybrid algorithm containing Learn^{++}.NSE and SMOTE for incremental learning in a nonstationary environment where the data distribution shows moderate to severe class imbalance. We have shown that this combination has the advantage of boosting the recall of the minority class both in synthetic data of controlled drift, as well as a real world application where the nature and rate of drift are not known. The results indicate that the recall of a minority class can be greatly improved in a nonstationary environment, with statistical significance, compared to the original Learn^{++}.NSE algorithm, which itself was shown to work very well on nonstationary environments whose data distribution did not suffer class imbalance [6;7;9].

Future work includes the removal or partial or complete removal of error term from the algorithm to make it a better candidate for class imbalance in a concept drift scenario by using different statistical measures like the f-measure, g-mean, precision and recall as well as comparing Learn^{++}.SMOTE to algorithms presented in [3;11]. Some future questions we are interested addressing include how injecting SMOTE into Learn^{++}.NSE algorithm affect different base classifiers, and the effect of adding SMOTE under different drifting conditions.

Acknowledgment

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

References

[1] 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.

[2] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: a new ensemble method for tracking concept drift," *3rd IEEE Int. Conf. on Data Mining (ICDM 2003)*, pp. 123-130, 2003.

[3] 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.

[4] K. Nishida, K. Yamauchi, and O. Takashi, "ACE: Adaptive Classifiers-Ensemble System for Concept-Drifting Environments," *Multiple Classifier Systems*in Lecture Notes in Computer Science, eds. N. Oza, R. Polikar, J. Kittler, and F. Roli, Eds., vol. 3541, pp. 176-185, 2005.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] M. B. Harries, "SPLICE-2 Comparative Evaluation: Electricity Pricing," The University of New South Wales, Sydney, Australia,1999.

[11] 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.