Incremental Learning in Non-stationary Environments with Concept Drift using a Multiple Classifier Based Approach

Matthew Karnick, Michael D. Muhlbaier and Robi Polikar*
Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA
karnick@ieee.org, muhlbaier@ieee.org, *Corresponding author: *polikar@rowan.edu*

Abstract

We outline an incremental learning algorithm designed for nonstationary environments where the underlying data distribution changes over time. With each dataset drawn from a new environment, we generate a new classifier. Classifiers are combined through dynamically weighted majority voting, where voting weights are determined based on classifiers' age and accuracy on current and past environments. The most recent and relevant classifiers are weighted higher, allowing the algorithm to appropriately adapt to drifting concepts. This algorithm does not discard prior classifiers, allowing efficient learning of potentially cyclical environments. The algorithm learns incrementally, i.e., without access to previous data. Finally, the algorithm can use any supervised classifier as its base model, including those not normally capable of incremental learning. We present the algorithm and its performance using different base learners in different environments with varying types of drift.

1. Introduction

One of the more challenging problems in pattern recognition is learning from a data distribution that experiences concept drift, otherwise known as a nonstationary environment. In a nonstationary environment (NSE), the underlying data distribution and the corresponding decision boundaries change over time. Such problems need an algorithm that can detect the concept drift, and adapt to consequent changes. Further complications arise if previously learned information is partially relevant, and the data corresponding to such information are no longer available. Such a scenario requires an *incremental learning* algorithm that provides a balance between stability and plasticity [1].

Traditionally, concept drift is addressed by training a new classifier with each new dataset, and then using that classifier until the environment changes again, as in FLORA family of algorithms [2,3]. More recently, multiple classifier system based algorithms have been proposed for nonstationary learning. Such algorithms generate and then combine an ensemble of classifiers, where each classifier is trained on a different snapshot of the data distribution. In her recent review, Kuncheva places ensemble based approaches into one of three categories [4]: (i) a fixed ensemble whose combination rules (weights) are changed based on the changing environment (dynamic combiners), as in Winnow [5]; (ii) an ensemble where new data is used to update the parameters or members of an online learning algorithm, as in Oza's online boosting [6]; and/or (iii) altering the structure of the ensemble by adding new members to an existing ensemble, such as Nishida's adaptive classifier ensemble (ACE) [7], or replacing the least contributing ensemble members with a new one generated based on new data, such as Street's streaming ensemble algorithm (SEA) and Kolter's dynamic weighted majority (DWM) [8,9].

We have recently introduced – and evaluated on normally distributed drifting data – an algorithm combining different aspects of these approaches [10]. Specifically, a new classifier is generated on each new dataset, and the classifiers are then combined based on their age and performance on current and past environments. The algorithm learns incrementally, i.e., without having access to previously available data. We do not make any assumption on the form of the drift: it can be deterministic or random, contracting or expanding, gradual, abrupt, or even cyclical. To accommodate a diverse set of environments, the algorithm retains every classifier created, so that it can refer to information acquired by – still relevant – former classifiers; or access previous knowledge that may later become relevant again. We call this algorithm Learn^{++}.NSE, based on our previously introduced incremental learning algorithm Learn^{++} [11].

In this paper, we outline Learn^{++}.NSE and evaluate its performance (using three different base classifiers) on two challenging scenarios, representing different types of drift (gradual, cyclical and abrupt changes). We show that the algorithm not only adapts well to concept drift, but also outperforms a single classifier that has access to the most recent data, regardless the type of base classifier used.

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

2. Learn^{++}.NSE

Learn^{++}.NSE is provided with a series of training datasets, each drawn from an unknown distribution that is the current snapshot of a possibly drifting environment. The algorithm is presented with each batch of data only once, hence incremental learning. Each batch of data is used to generate a new classifier, which are then combined through weighted majority voting (WMV). The voting weights are computed as a weighted average of each classifier's errors at all time steps, where current environments are more heavily weighted. Hence, the algorithm tracks concept drift by two mechanisms: new classifiers trained on each dataset, and the dynamically updated voting weights. Note that if older classifiers begin to perform well on new data, they are awarded with higher voting weights.

The algorithm has two inputs: a supervised algorithm, BaseClassifier, to train individual classifiers, and the current training data at time *t*. Learn^{++}.NSE maintains a distribution $D^t(i)$ over the training data instances $\boldsymbol{x_i}$. $D^1(i)$ is set to be uniform, giving equal probability to each instance being drawn from the first dataset. When new data arrive at time *t*, one new classifier h_t is generated and combined with all previous classifiers to create the composite hypothesis H_t. The decision of H_t is the ensemble decision. Before a new h_t is trained, however, Learn^{++}.NSE first evaluates the ensemble's knowledge of the current environment through the error E^t of the current H^{t-1} on the new dataset, which is used to update the distribution weights (Steps 1 and 2 within the Do loop in Figure 1). The distribution update increases the probability of drawing instances that have not been correctly learned yet.

The algorithm then calls the BaseClassifier to create h_t using data drawn from the current training dataset (Step 3). All classifiers generated thus far h_k, k=1,...,t are evaluated on the current dataset, by computing ε_k^t, the error of the k^{th} classifier h_k at time t (Step 4), which gives us t error values - one for each classifier generated thus far. If the error of the most recent classifier on its own training data is greater than ½, we discard that classifier and generate a new one, as it will most likely make no positive contribution to the ensemble. If the error of older classifiers are greater than ½, they are set to ½ and we do not discard them. This effectively sets the normalized error of that classifier to 1 (Equation 5), removing its voting power (Equation 7), but *at that time step* only; older classifiers may later obtain higher voting weights, if they perform well on a future environment's data.

To determine the voting weights, we first obtain a weighted average of all t error measures ε_k^t for each classifier h_k. A nonlinear sigmoid function $f(\cdot)$ is used for the weighting (Equation 6), which essentially weights newer environments more heavily.

Input: For each dataset $\mathfrak{D}^t$ $t = 1,2,\ldots$
Training data $\{\boldsymbol{x_i^t} \in X; y_i^t \in Y = \{1,\ldots,c\}\}, i = 1,\cdots,m^t$, and supervised algorithm **BaseClassifier**
Do for $t = 1,2,\ldots$
If $t = 1$, **Initialize** D^1 to be uniform; **Go to** step 3.
Endif

1. Compute current ensemble error on new data
$$E^t = \sum_{i=1}^{m^t} (1/m^t) \cdot [\![H^{t-1}(x_i) \neq y_i]\!] \quad (1)$$

2. Update and normalize instance weights
$$w_i^t = \frac{1}{m^t} \cdot \begin{cases} E^t, H^{t-1}(x_i) = y_i \\ 1, otherwise \end{cases} \quad (2)$$
$$\text{Set } D^t = w^t / \sum_{i=1}^{m^t} w_i^t \implies D^t \text{ is a distribution} \quad (3)$$

3. Call **BaseClassifier** with $\mathfrak{D}^t$, obtain $h_t: X \to Y$

4. Evaluate all existing classifiers on new data $\mathfrak{D}^t$
$$\varepsilon_k^t = \sum_{i=1}^{m^t} D^t(i) \cdot [\![h_k(x_i) \neq y_i]\!] \text{ for } k = 1,\ldots,t \quad (4)$$
If $\varepsilon_{k=t}^t > 1/2$, generate a new h_t.
If $\varepsilon_{k<t}^t > 1/2$, set $\varepsilon_k^t = 1/2$,
$$\beta_k^t = \varepsilon_k^t/(1-\varepsilon_k^t), \text{ for } k = 1,\ldots,t \quad (5)$$

5. Compute the sigmoid $f(\cdot)$ weighted sum of all normalized errors for k^{th} classifier h_k:
$$\bar{\beta}_k^t = \sum_{j=0}^{t-k} f\left(\beta_k^{t-j}\right), \text{ for } k = 1,\ldots,t \quad (6)$$

6. Calculate classifier voting weights
$$W_k^t = \log\left(1/\bar{\beta}_k^t\right), \quad for\ k = 1,\ldots,t \quad (7)$$

7. Obtain the final hypothesis
$$H^t(x_i) = \arg\max_c \sum_k W_k^t \cdot [\![h_k(x_i) = c]\!] \quad (8)$$

Figure 1: Learn^{++}.NSE Algorithm

This process gives less weight *not* to old classifiers, but to their *error on old environments*. Therefore, a classifier generated long time ago can still receive a large voting weight, if its error on the *recent* environments is small. Finally, all classifiers are combined through weighted majority voting (Step 7, Equation 8).

3. Simulations and Results

We have evaluated this algorithm on two challenging non-stationary datasets. The first dataset is the two-class two-feature rotating checkerboard (see Figure 2 for six snapshots), where we vary the rotational parameter α as a function of time. As α increases, the checkerboard rotates clockwise from t=0 to t=1 in 700 steps, making one complete rotation. The class distribution repeats itself at $\alpha = \pi$ and 2π, hence we expect that classifiers generated during $\alpha = [0, \pi]$ interval to be relevant again in the $\alpha = [\pi, 2\pi]$ interval. Figure 2 shows the training data (merely 50 instances from each class) – plotted in oversized circles for clarity – and overlaid on the densely sampled 2601-instance (51x51 grid) test data clearly showing the decision boundaries.

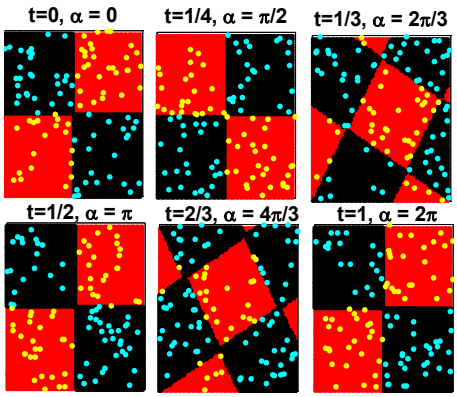

Figure 2. Rotating checkerboard data snapshots.

Random noise was also added to each feature to prevent presenting the algorithm with duplicate snapshots during the $\alpha = [\pi, 2\pi]$ interval.

Our second experiment is benchmark dataset, *SEA concepts* [7], where abrupt concept drifts are present. The SEA concepts consist of randomly generated three-feature data partitioned into four concepts. The corresponding class label during each concept is a function of the first two features only: class one if the sum of the two features exceeds a certain threshold, class two otherwise: $c_i = 1$ if $f_{i,1} + f_{i,2} \geq \theta_t$, and $c_i = 2$ otherwise. This essentially generates a plane through the three-dimensional space. The threshold θ controls the changing concepts. We define the time axis to begin at $t = 0$ and complete at $t = 1$; and as in [9], $\theta_t = 8$ for $t = 0$ to 0.25; 9 for $t = 0.25$ to 0.5; 7 for $t = 0.5$ to 0.75, and 9.5 for $t = 0.75$ to 1. We have also added a 10% class noise to the training data as in [9].

We repeated all experiments using three different classifiers types to evaluate the algorithm's ability to work with different models: multilayer perceptron (MLP, 25 hidden layer nodes, 0.1 error goal), support vector machine (SVM, Gaussian kernel, $\sigma = 0.5$, $C = 10{,}000$) and naïve Bayes (NB). Finally, we benchmarked the performance of Learn^{++}.NSE against the single classifier (of identical parameters) that is most recently added to the ensemble. Note that this is not a trivial comparison, as the last classifier is expected to outperform every classifier in the ensemble trained on previous environments. For an ensemble to outperform this single classifier, each individual member must contribute knowledge applicable to the current environment, despite not having seen any (current) data.

3.1 Results on Checkerboard Data

Figure 3 shows the ensemble performance compared to that of current single classifier at each time step, for all three types of base classifiers. We make several observations from this figure. First, the ensemble performance is always as good as or better than the most recent single classifier, indicating that the algorithm can indeed track the concept drift. Second, the

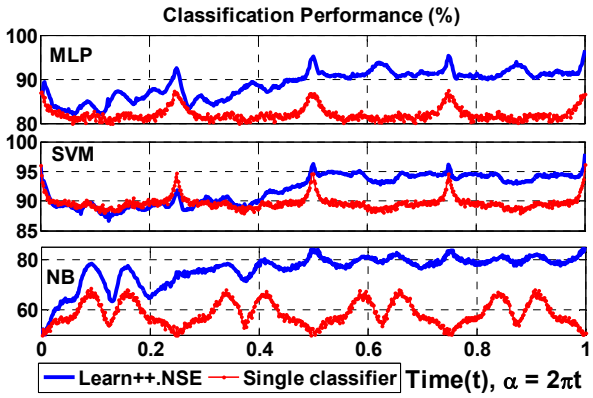

Figure 3. Rotating checkerboard performances.

ensemble performance increases sharply after $t = 0.5$, as class distributions then repeat and Learn^{++}.NSE takes advantage of its previously trained classifiers. Previously generated classifiers are now reweighted and available to contribute to the algorithm's performance, a true benefit of the algorithm's policy on retaining all classifiers for cyclic environments. Also note the sharp performance increases at $t = \{0.25, 0.5, 0.75, 1\}$: at these instances, the checkerboard is rotated at a right angle, resulting in decision boundaries being substantially simpler compared to those rotated at other angles.

3.2 Results on SEA Concepts

The SEA concepts dataset introduced by Street et al. [9] is a commonly used benchmark for nonstationary environments. The data consists of 50,000 instances (12,500 for each concept), introduced to the algorithm in blocks of 250 instances.

Figure 4 presents the results obtained by three types of base classifiers. Throughout the simulation, Learn^{++}.NSE continually outperformed the single classifier. We observe that the single classifier, always trained on the most recent data, retains a constant performance, whereas Learn^{++}.NSE gradually improves its performance. The time instances $t = 0.25, 0.5, 0.75,$

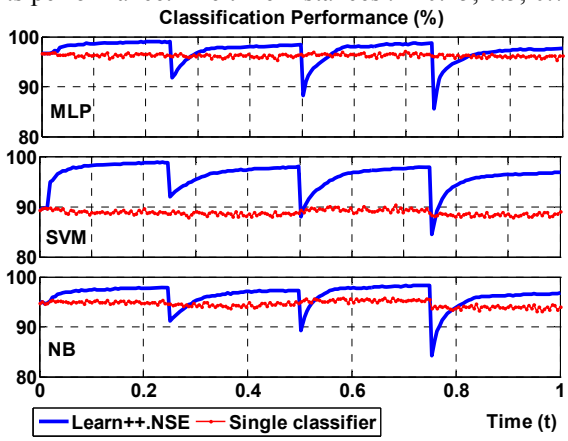

Figure 4. SEA concepts performances

where the abrupt concept drift occurs is clearly visible. The algorithm detects the concept drift (a temporary dip in performance) at these locations, and it quickly recovers by adapting to the new environment, and exceeds the performance of the single classifier. As more experts trained on the new concept are added to the ensemble, the algorithm is able to weigh them appropriately, and quickly achieve its optimal performance level. We also note that the algorithm works equally well with all types of classifiers, whether normally capable of online or incremental learning (such as naïve Bayes) or not (such as SVM or MLP).

4. Discussions and Conclusion

We have described Learn^{++}.NSE, an ensemble of classifiers based incremental learning algorithm, capable of learning from concept drift in non-stationary environments. As new datasets are presented, Learn^{++}.NSE adds new classifiers to the existing ensemble and adjusts voting weights to combine these classifiers through weighted majority voting. The novelty of the algorithm primarily rests on its ability to dynamically adjust voting weights based on the performance of all classifiers on current and past environments. The algorithm does not store or require access to the previous data, and each classifier is trained only on the currently available data. If the knowledge acquired by older classifiers become irrelevant, Learn^{++}.NSE temporarily reduce / annul their voting weights as appropriate - rather than discarding them. Note that since we cannot predict the future behavior of a nonstationary environment ahead of time, a given classifier's complete obsoleteness cannot be justified – hence Learn^{++}.NSE avoids discarding classifiers, providing a delicate stability-plasticity balance.

On two challenging nonstationary environments (one cyclical and gradual change, the other with very abrupt concept changes), we have shown that the algorithm is able to track both types of drift very well, regardless of the type of base classifier used. Specifically, on the rotating checkerboard problem, we showed that the algorithm can track gradually drifting concepts, where each added classifier further improves the algorithm accuracy – despite the environment continuously changing at each time step. The benefit of retaining all classifiers was also demonstrated on this dataset: after one complete rotation of the class distributions, the algorithm significantly improved its performance, with wide margins above that of a single classifier trained on the current dataset. The ability of the algorithm to recover from instantaneously abrupt changes was then demonstrated on the SEA concepts data: with the addition of just a few classifiers, the algorithm quickly learned the new concepts.

While this algorithm is not intended for all concept drift problems, we expect Learn^{++}.NSE to perform well on a variety of concept drift scenarios, and particularly well i) when each individual classifier is presented with limited amount of data that cannot give an adequate representation of the current environment, ii) where previously trained classifiers still carry at least some amount of relevant information, or iii) where the environment changes in a cyclical manner.

Although Learn^{++}.NSE is still in development, preliminary results are very promising and warrant further analysis of this approach. Future work includes testing the algorithm with higher dimensional real world datasets, and/or problems where the rate of change itself changes in time.

Acknowledgements

This material is based upon work funded by the National Science Foundation grant No: ECS-0239090.

References

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

[2] 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.

[3] R. Klinkenberg, "Learning Drifting Concepts: Example Selection vs. Example Weighting," *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281-300, 2004.

[4] L. I. Kuncheva, "Classifier Ensembles for Changing Environments," *Multiple Classifier Systems (MCS 2004)*, LNCS vol. 3077, 2004, pp. 1-15.

[5] A. Blum, "Empirical Support for Winnow and Weighted-Majority Algorithms" *Machine Learning*, vol. 26, no. 1, pp. 5-23, Jan.1997.

[6] N. Oza, "Online Ensemble Learning." Ph.D. Dissertation, University of California, Berkeley, 2001.

[7] K. Nishida, K. Yamauchi, O. Takashi, "ACE: Adaptive Classifiers-Ensemble System for Concept-Drifting Environments," LNCS vol. 3541, 2005, pp. 176-185.

[8] J. Z. Kolter, M. Maloof, "Dynamic weighted majority: a new ensemble method for tracking concept drift," *IEEE Int. Conf. Data Mining*, 2003, pp. 123-130.

[9] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," *Knowledge Discovery & Data Mining*, 2001, pp. 377-382.

[10] M. Muhlbaier, R. Polikar, "Multiple Classifiers Based Incremental Learning Algorithm for Learning in Nonstationary Environments," *Int. Conf. Machine Learning and Cybernetics,* vol. 6, 2007, pp. 3618-3623.

[11] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Sys. Man, Cybernetics* (*C*), vol. 31, no. 4, pp. 497-508, 2001.