Incremental Learning of Variable Rate Concept Drift

Ryan Elwell and Robi Polikar*

Signal Processing and Pattern Recognition Laboratory
Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA
ryan.elwell@ieee.org, polikar@rowan.edu

Abstract. We have recently introduced an incremental learning algorithm, Learn^{++}.NSE, for *N*on-*S*tationary *E*nvironments, where the data distribution changes over time due to concept drift. Learn^{++}.NSE is an ensemble of classifiers approach, training a new classifier on each consecutive batch of data that become available, and combining them through an age-adjusted dynamic error based weighted majority voting. Prior work has shown the algorithm's ability to track gradually changing environments as well as its ability to retain former knowledge in cases of cyclical or recurring data by retaining and appropriately weighting all classifiers generated thus far. In this contribution, we extend the analysis of the algorithm to more challenging environments experiencing varying drift rates; but more importantly we present preliminary results on the ability of the algorithm to accommodate addition or subtraction of classes over time. Furthermore, we also present comparative results of a variation of the algorithm that employs an error-based pruning in cyclical environments.

Keywords: nonstationary environment, concept drift, Learn^{++}.NSE.

1 Introduction

Classification in changing environments is a particularly interesting and challenging problem with a growing list of application domains, such as network monitoring, economic, climate or financial data analysis, all of which generate data from such environments. The complexity of the problem is in part due to decision boundaries between classes being subject to potentially unpredictable change over time, which is commonly referred to as a non-stationary environment, and is most commonly associated with a change or *drift* in the underlying distribution of the data. The ability to track such data requires an algorithm that can update its parameters such that new knowledge is acquired, while old – and still relevant – information is retained, but the currently irrelevant information is discarded only to be recalled if such data later become relevant again in a cyclical environment. Ideally, such an algorithm should be incremental, i.e., not requiring access to previously seen data.

Earliest efforts in learning concept drift have focused on the types of drift and conditions under which such drift can be learned, e.g. under a formal PAC framework, as a hidden context [1;2], or more pragmatically as real vs. virtual drift [3]. More recently, Kuncheva indicated four different types of changes that may be encountered,

* Corresponding author.

J.A. Benediktsson, J. Kittler, and F. Roli (Eds.): MCS 2009, LNCS 5519, pp. 142–151, 2009.
© Springer-Verlag Berlin Heidelberg 2009

such as noise (which should ideally be recognized as such, and ignored by the algorithm), a blip (a rare event, anomaly detection), an abrupt or gradual change [4;5].

Various approaches have been developed for concept drift, which we categorize as *active* or *passive* approaches, also referred to as *detect-and-retrain* and *constant update* [6]. An active approach explicitly seeks to detect drift in the data, and takes corrective action to keep the classifier up-to-date. FLORA is among the first such algorithms, and uses an adjustable window on the incoming data to ensure that only data from the current environment is used for classifier training [2]. The window size is based on a performance on current environment, which is also interpreted as a measure of the amount of drift. Any data that fall outside of the current window are then assumed to be irrelevant and the corresponding knowledge is forgotten by replacing the existing classifier with the newly trained one. In most active parameter or performance based drift detection approaches, the environment is assumed to be stationary unless some parameter (e.g. difference of certain statistical moments between prior and incoming data, [7-9] or performance threshold [10] is surpassed.

Passive approaches to tracking drift operate under the assumption that drift is always occurring, thus avoiding the complexity of trying to determine the magnitude of change. Therefore, the same procedure – e.g., using a fixed window size or always retraining a new classifier – is followed for each instance or batch of incoming data.

Ensemble or multiple classifier systems (MCS) based algorithms, which are typically passive approaches, represent a new breed of algorithms to learning in non-stationary environments, and they are particularly effective at providing a good balance between stability (retaining existing and relevant information) and plasticity (learning new knowledge) in the presence of drift. MCS-based approaches consist of an ensemble of classifiers, combined to form a final representative decision. In order to prevent irrelevant knowledge from effecting this decision, a combination of voting techniques and forgetting mechanisms are employed. Voting based classifier combination (ensemble weighting) allows classifiers with varying competences on the current environment to proportionally contribute to the final decision as in dynamic majority voting (DWM) [10], LIFT-based weight assigning [11], adaptive classifiers for changing environments (ACE) [12] or others [6;13]. Weights are often dynamically updated at each training instance, independent of the existence or amount of drift. While lower weights allow the knowledge of certain classifiers to be temporarily forgotten, *ensemble pruning* allows knowledge believed to be completely irrelevant to be permanently discarded.

Pruning can be useful because the ensemble, growing otherwise uncontrollably over time with new classifiers trained on new data, may accumulate too many irrelevant classifiers that can outvote the competent experts in the ensemble, despite weighting strategies. Pruning also helps with another concern with ensemble approaches, namely, the computational complexity that increases with each new classifier generated for incoming data. The preventative approach taken in many algorithms is therefore to limit the number of classifiers in the ensemble using an error-based criterion, where the worst performing expert in the ensemble is permanently removed. This has been shown to be an effective method in many studies, and has proven superior to the more basic age-based pruning (remove the oldest) [10;12-14]. We must note, however, that permanent removal of classifiers through pruning runs the risk of discarding information that may later become relevant, should the environment happen to be a cyclical

one. The trade-off between using a pruning approach and retaining all classifiers is analyzed in this effort.

The aforementioned approaches have been evaluated on a variety of concept drift scenarios, such as gradual or abrupt, but not on cyclical environments, except our current and previous efforts [15-17], and certainly not on environments where the rate of change itself changes unpredictably, or where the change introduces or removes classes. In this contribution, we evaluate and compare our proposed approach, Learn^{++}.NSE, and its error-based pruning version on such challenging environments.

2 Learn^{++}.NSE

Learn^{++}.NSE is an ensemble-based (passive) approach for *N*on*S*tationary *E*nvironments. Data are drawn in batches over time from an environment that may be experiencing some change; however, we make no assumption on the nature or rate of the drift. The change – if and when it exists – may be gradual or abrupt, expanding or contracting in feature space, introducing or removing classes, cyclical or otherwise. We simply assume that the current data distribution has changed in some way compared to the prior distribution that provided the previous data. For each consecutive batch or snapshot of the data, a new classifier is generated. In the original version of Learn^{++}.NSE, all classifiers are retained, not only to maintain ensemble stability, but also to accommodate cyclic environments. To reduce outvoting from an eventually large number of potentially irrelevant classifiers, Learn^{++}.NSE tracks the age-weighted running average of the ensemble error over all current and previous environments, which are then used to determine classifier voting weights. A sigmoidal weighting gives most recent error a greater consideration for determining the classifier weights – regardless of the age of the classifier. Note that it is the error of the classifiers on more recent environments that are weighted more heavily, and not the classifiers themselves. Hence old classifiers may receive higher current weights.

The pseudocode in Figure 1 formally describes Learn^{++}NSE. Data are received as snapshots $\mathfrak{D}^t$ of the current environment with distribution $P^t(x, y)$. At each time step, m^t samples are drawn as the training dataset $\{\boldsymbol{x_i^t} \in X; y_i^t \in Y\}, i = 1, \cdots, m^t$. A Base Classifier is used to train a classifier on the current dataset. Each consecutive snapshot $\mathfrak{D}^{t+1}$ is assumed to be drawn from distribution $P^{t+1}(x, y)$ that differs from that of the prior snapshot. Moreover, we assume that prior data is no longer accessible; thus the information must be stored solely within the classifiers. This assumption is a necessary condition for incremental learning algorithms.

When new data arrive at time *t*, Learn^{++}.NSE first evaluates the overall error ε_k^t of the existing ensemble of *k* classifiers on the new training data (Step 1). A normalized distribution D^t is adjusted – similar to that in AdaBoost [18] – according to the errors committed on each instance. This distribution is greater over those points at which the composite hypothesis $H^{t-1}(x_i)$ does not match the true class label (Step 2). This distribution D^t is an important tool to be used for individual classifier error evaluation. Once a new classifier is added to the ensemble (Step 3), the errors of all classifiers – normalized with respect to D^t – are computed on the new data $\mathfrak{D}^t$. Since the distribution D^t gives more emphasis to previously misclassified instances, the current error gives classifiers more credit for correctly classifying instances on which the ensemble

hypothesis was incorrect. Note that if the newly trained classifier's error on the current environment exceeds ½, that classifier is discarded and a new one is trained; if the error of a previously generated classifier exceeds ½, however, its weight is set to ½. An error of ½ yields a normalized error β_k^t of 1 (Equation 6) at the current time step, which then carries zero voting weight (Equation 9).

The classifier errors over all (current and previous) environments are then averaged using a sigmoidal weighting function, giving more weight to errors on more recent environments (step 5). This allows retention and *reactivation* of old classifiers if they perform well on new environments. If the ensemble size exceeds a predetermined threshold T, and if pruning is desired, then the classifier with the largest error on the current data is discarded (step 6). The average error is then used to determine the voting weight of each classifier (step 8), combined through weighted majority voting (step 9).

Input: For each dataset $\mathfrak{D}^t \quad t = 1,2, \ldots$
Training data $\{\boldsymbol{x}_i^t \in X; y_i^t \in Y = \{1, \ldots, c\}\}, i = 1, \cdots, m^t$.
Supervised learning algorithm **BaseClassifier**
Ensemble size T
Do for $t = 1,2, \ldots$

If $t = 1$, **Initialize** $D^1(i) = w^1(i) = 1/m^1$, $\forall i$, Go to step 3. **Endif** (1)

1. Compute error of the existing ensemble on new data
   $E^t = \sum_{i=1}^{m^t} (1/m^t) \cdot [\![H^{t-1}(x_i) \neq y_i]\!]$ (2)
2. Update and normalize instance weights
   $w_i^t = \frac{1}{m^t} \cdot \begin{cases} E^t, H^{t-1}(x_i) = y_i \\ 1, otherwise \end{cases}$, (3)
   Set $D^t = w^t / \sum_{i=1}^{m^t} w_i^t \implies D^t$ is a distribution (4)
3. Call **BaseClassifier** with $\mathfrak{D}^t$, obtain $h_t: X \to Y$
4. Evaluate all existing classifiers on new data $\mathfrak{D}^t$
   $\varepsilon_k^t = \sum_{i=1}^{m^t} D^t(i) \cdot [\![h_k(x_i) \neq y_i]\!]$ for $k = 1, \ldots, t$ (5)
   If $\varepsilon_{k=t}^t > 1/2$, generate a new h_t. **If** $\varepsilon_{k<t}^t > 1/2$, set $\varepsilon_k^t = 1/2$,
   $\beta_k^t = \varepsilon_k^t / (1 - \varepsilon_k^t)$, for $k = 1, \ldots, t$ (6)
5. Compute weighted average of all normalized errors for k^{th} classifier h_k:
   For $a, b \in \mathbb{R}$, $\omega_k^t = 1/\left(1 + e^{-a(t-k-b)}\right)$, $\omega_k^t = \omega_k^t / \sum_{j=0}^{t-k} \omega_k^{t-j}$ (7)
   $\bar{\beta}_k^t = \sum_{j=0}^{t-k} \omega_k^{t-j} \beta_k^{t-j}$, for $k = 1, \ldots, t$ (8)
6. Ensemble Pruning (if used): **If** $t > T$
   Error-based: Remove h_k where $\varepsilon_k^t = \max_{k=1 \ldots t} \varepsilon_k^t$
   Endif
7. Calculate classifier voting weights $W_k^t = \log(1/\bar{\beta}_k^t), \; for \; k = 1, \ldots, t$ (9)
8. Obtain final hypothesis: $H^t(x_i) = \arg\max_c \sum_k W_k^t \cdot [\![h_k(x_i) = c]\!]$ (10)

Fig. 1. Learn^{++}.NSE Algorithm pseudocode

3 Experimental Results

Previous studies have shown that Learn^{++}.NSE i) consistently outperforms a single classifier, justifying an MCS approach; ii) is able to poll old classifiers in cases of recurring or cyclical data and consequently achieve higher accuracy; and iii) can work with a wide spectrum of on-line as well as batch learning algorithms, and hence is independent of the base classifier used to train the algorithm [15-17]. In this study, we pose the problem of even harsher environments which involve changes in the *number of classes* and changes in the *rate of drift* within the experiment (acceleration of drift). We also evaluate the trade-off made by pruning the ensemble and the ability to recall previously seen knowledge in recurring environments.

In prior work we have used a unique non-Gaussian dataset, the rotating checkerboard, derived from the canonical XOR problem. Figure 2 shows several snapshots of the distribution from which the data are drawn. A static window is maintained for sampling at each time step while the checkerboard itself rotates about a central axis. In each complete rotation ($\alpha = 0$ to 2π), the distribution inside the window repeats twice, yielding a recurring context. Random noise is introduced at each time step to prevent identical snapshots of training data from appearing. We also use a minimal number of training data (size m=25) to test the limit of the algorithm.

In this experiment, we compare the Learn^{++}.NSE performance during an accelerating (positive or negative) drift. We introduce three cases as described in Figure 3: a basic constant drift, an exponentially increasing drift, and a sinusoidal drift rate which includes a momentary pause at the midpoint of the test. From the start (t = 0) to finish (t=1), the board completes one rotation ($\alpha = 0$ to 2π) in T = 400 time steps, according the rate of changes shown in Fig. 3. The purpose of these tests is to investigate the algorithm's ability to track harsh environments while still maintaining the characteristics that have been realized thus far: the algorithms ability to significantly and consistently outperform a single classifier trained on the current training data (justifying an ensemble of classifiers), the ability to track recurring or cyclical data (justifying the error-weighting approach to prevent outvoting), and the ability to achieve a performance better than a pruned ensemble (justifying the retention of classifiers).

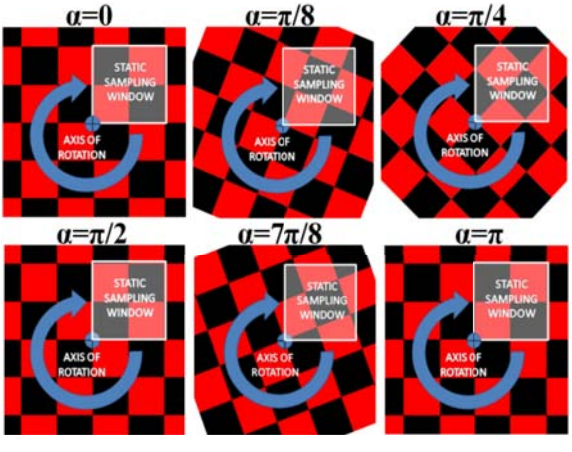

Fig. 2. Rotating checkerboard data

Results are computed on the entire testing grid of 51x51 resolution (2601 test points). All plots are averages of 100 independent trials and include a 95% confidence interval in a similarly colored shaded region around each curve (please see the electronic version of this paper for optimal viewing of the colors). For each experiment,

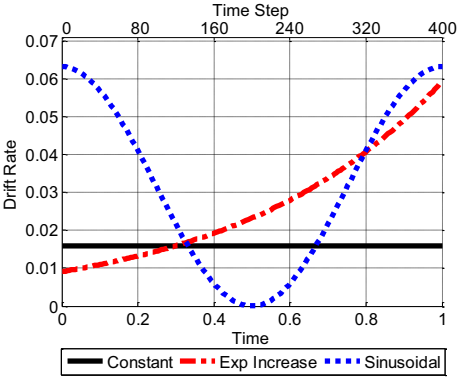

Fig. 3. Variable drift rates

we compare Learn^{++}.NSE to a single sifier and to Learn^{++}.NSE with error-based pruning, where a 25-classifier cap is maintained. Separate experiments are conducted for each dataset with different base classifiers including naive Bayes, MLP, and SVM. For brevity, and since all three yielded similar trends, only SVM results are given here.

Results in Figures 4 ~ 6 appear to be quite promising despite the changes in the amount and rate of drift throughout the experiment. First, Learn^{++}.NSE (with or without pruning) consistently outperforms a single classifier with statistical significance, regardless of the rate or type of the drift. This is not a trivial, but an important benchmark: a single classifier is always trained on the most recent data only, and never has to deal with former knowledge. The performance of Learn^{++}.NSE over a single classifier indicates that the algorithm can extract useful information from the previous classifiers, and that the ensemble structure is in fact beneficial. Second, the ability of the algorithm to track the changing the environment, as expected, is correlated to the current rate of change: when the environment is changing very slowly or when it is stationary, for example, during the mid sections of the sinusoidal rate of change (Fig. 6), the ensemble performance increases very rapidly. When the rate of change is accelerating (e.g. after 300 ms in Fig. 5, or 200 ms in Fig. 6), there is a slight drop in the performance; and when the environment is changing at a constant rate, the algorithm performance increases gradually (Fig 4.) We should note here that the sharp performance peaks in Fig 4 ~ 6 are simply due to the periodic nature of the problem, where decision boundaries become perpendicular (and therefore simpler) every $\pi/2$ radians. Also note that the current state of the board is different at any given time step for each of the experiments due to varying rates of change.

Perhaps one of the most interesting observations is the behaviour of the algorithm at and after $\alpha = \pi$. This happens at time step T = 200, 275 and 200 for the constant, accelerating and sinusoidal change of rate, respectively. As mentioned above, the distribution repeats itself in the $\alpha = [\pi \sim 2\pi]$ interval, creating a cyclic environment. Learn^{++}.NSE (both the pruned and unpruned versions) shows an increase in performance after this interval, compared to $\alpha = [0 \sim \pi]$ interval, a clear indication of the ability of the algorithm to use old classifiers which then become relevant again. This is particularly striking in Figure 5, where the performance improvement due to using old classifiers outweighs the performance drop due to rapidly accelerating rate of change which also occurs at around T =275 ($\alpha = \pi$). Finally, upon careful observations, we also observe that the unpruned Learn^{++}.NSE consistently and significantly outperforms the pruned version in the $\alpha = [\pi \sim 2\pi]$ interval (Fig. 4 and 5), indicating that those classifiers discarded by the pruning can in fact be useful in such a cyclic environment. The only exception to this is in Fig 6, where the large number of accumulated classifier during very slow / near stationary period of the sinusoidal rate does not allow the unpruned ensemble to significantly outperform the pruned ensemble for

a short period of time (T=275) when the rate of change starts increasing again. It is interesting to note that during the very end (at around T = 400, or $\alpha = 2\pi$), both versions of Learn^{++}.NSE – but in particular the unpruned version – shows a more significant performance peak. This is attributed to the return of a prior distribution, now for the third time, allowing a larger number of classifiers to contribute to a correct decision.

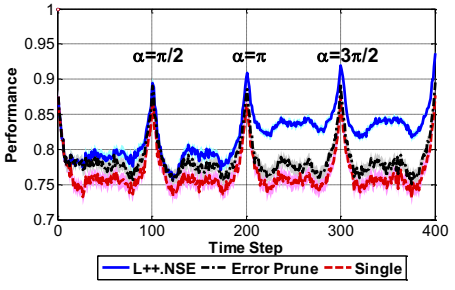

Fig. 4. SVM performance on checkerboard with constant drift rate

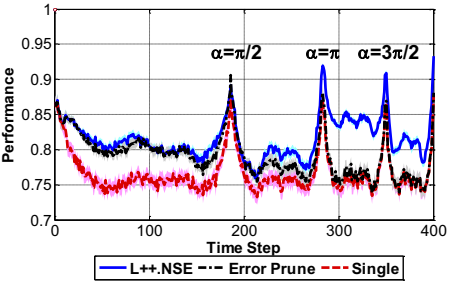

Fig. 5. SVM performance on checkerboard with exponentially accelerating drift rate

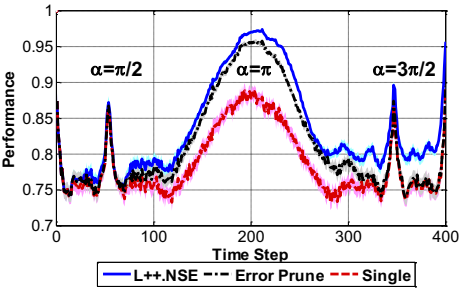

Fig. 6. SVM performance on checkerboard with sinusoidal drift rate

Hence, apart from validating that the algorithm can handle varying rates of drift, these experiments also allow us to determine when pruning may be beneficial: if the environment changes (relatively) at a constant rate, and/or the environment is not expected to repeat itself, then an error-based pruning provides performance nearly as good as the entire ensemble. However, in a cyclic environment (such as climate, electricity demand, etc. applications), using an unpruned ensemble is always preferable because of the sheer significance in performance increase.

The second experiment is a synthetic multi-class Gaussian dataset that includes random concept drift in the mean, variance, as well as the number of classes present at any given time. Figure 7 provides a visual display of the drift path followed by each class during the experiment, where the arrows indicate the direction of drift in mean for the next interval, and the sizes of the ellipses / circles indicate the variance of that class. Note that this scenario includes appearance of a new class (C_4) during

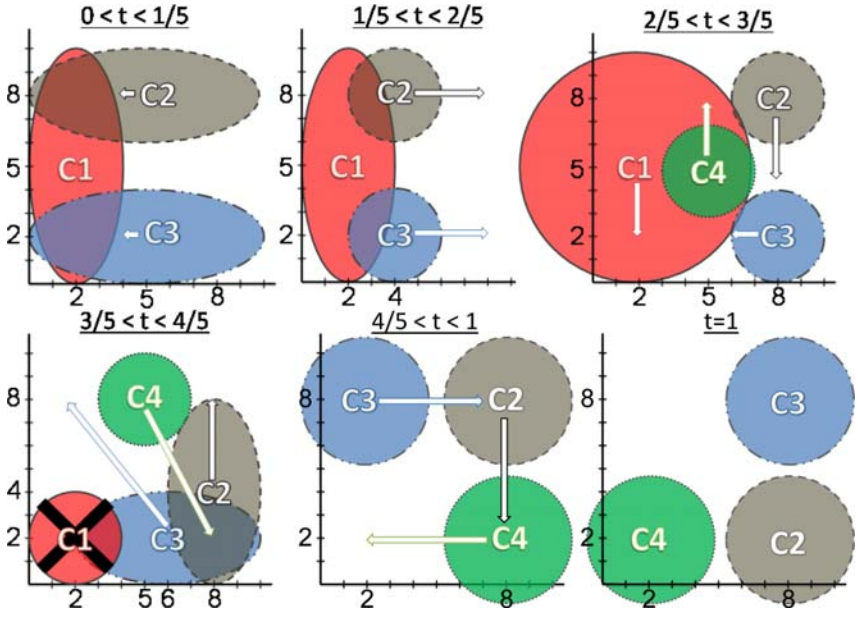

Fig. 7. Path of drift of multiclass Gaussian data from start at $t = 0$ to end at $t = 1$

$2/5 < t < 3/5$ and the disappearance of an old class (C_1) during $3/5 < t < 4/5$. At each time step in the $0 < t < 1$ interval, a new snapshot (size $m = 20$) of training data are obtained to train a new classifier for a total of T=300 time steps (snapshots).The performances of Learn^{++}.NSE (with and without pruning), a single classifier trained on the current training data, and that of ideal Bayes classifier (since the data distributions are Gaussian) are shown in Fig. 8 as averages of 100 independent trials along with their corresponding 95% confidence intervals (except for Bayes). The figure also indicates each separate interval of drift for reference.

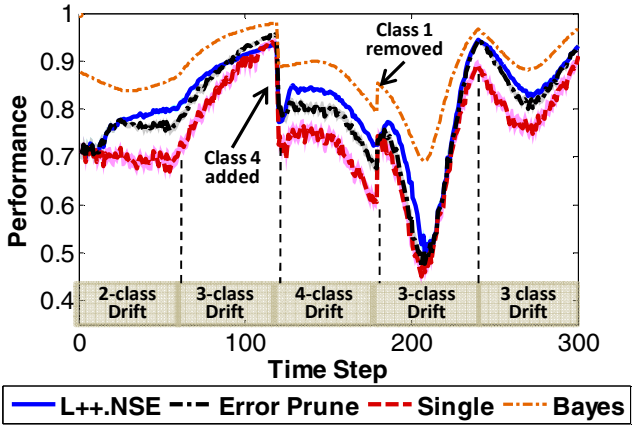

Fig. 8. Performance on the multiclass drift Gaussian data

The performance curve in Fig. 8 indicates that Learn^{++}.NSE and its pruned version are both capable of tracking an environment that experiences both concept drift and concept change (addition / removal of classes). The jumps in performance at T=120 and T=180 correspond to the sharp concept changes after the respective addition or removal of a class, which results (depending on the location of the class) in a change in the difficulty of the problem. We note that i) both versions of the algorithm follow the Bayes classifier very closely – hence able to track the drift, ii) there is little significant difference between the pruned and unpruned versions; and iii) they both outperform a single classifier, even (and especially) after class addition and subtraction.

4 Conclusions and Discussions

We presented an ensemble-based approach to the increasingly relevant topic of classification in non-stationary environments. We survey the complexity of this problem, as concept drift or change can occur in many forms (gradual, abrupt, cyclic, change in class counts, etc.) and rates (constant, increasing, decreasing, etc.). We show that Learn^{++}.NSE can closely follow and accommodate the drift, even in the harshest environments, regardless of its rate and type, or addition or removal of concept classes.

We have also compared the original version of Learn^{++}.NSE that retains all classifiers in case a cyclic environment makes the old classifier relevant to one that prunes the ensemble based on each classifier's error. The benefit of retaining all classifiers becomes especially evident when in fact an environment experiences a cyclic behavior, as the unpruned version significantly outperforms the pruned ensemble. However, in other cases, particularly when the rate of change is gradual and/or constant in rate, there is little or no significant difference between the two versions, and both versions significantly outperforms a single classifier trained on the most recent data.

Other attributes of the algorithm include independence of the base classifier being used, as well as being a truly incremental algorithm. At no point in time does Learn^{++}.NSE uses previously seen data and solely relies on the knowledge it extracts from the existing classifiers. Future work with Learn^{++}.NSE will include a comparison with other (active and passive) approaches on a variety of learning scenarios.

Acknowledgments

This material is based on work supported by the National Science Foundation under Grant No: ECS 0239090.

References

[1] Schlimmer, J.C., Granger, R.H.: Incremental Learning from Noisy Data. Machine Learning 1(3), 317–354 (1986)

[2] Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)

[3] Tsymbal, A.: Technical Report: The problem of concept drift: definitions and related work, Trinity College, Dublin, Ireland,TCD-CS-2004-15 (2004)

[4] Kuncheva, L.I.: Classifier Ensembles for Changing Environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)
[5] Kuncheva, L.I.: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: European Conference on Artificial Intelligence (ECAI), pp. 5–10 (2008)
[6] Rodriguez, J.J., Kuncheva, L.I.: Combining Online Classification Approaches for-Changing Environments. In: International Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition S+SSPR (2008)
[7] Alippi, C., Roveri, M.: Just-in-Time Adaptive Classifiers;Part I: Detecting Nonsta-tionary Changes. IEEE Transactions on Neural Networks 19(7), 1145–1153 (2008)
[8] Da Silva, B.C., Basso, E.W., Bazzan, A.L.C., Engel, P.M.: Dealing with non-stationary environments using context detection. In: 23rd International Conference on Machine Learning - ICML 2006, vol. 2006, pp. 217–224 (2006)
[9] Oommen, B.J., Rueda, L.: Stochastic learning-based weak estimation of multino-mial random variables and its applications to pattern recognition in non-stationary environments. Pattern Recognition 39(3), 328–341 (2006)
[10] Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: an ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
[11] Scholz, M., Klinkenberg, R.: Boosting Classifiers for Drifting Concepts. Intelligent Data Analysis, Special Issue on Knowledge Discovery from Data Streams 11(1), 3–28 (2007)
[12] Nishida, K., Yamauchi, K.: Adaptive Classifiers-Ensemble System for Tracking Concept Drift. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3607–3612 (2007)
[13] Wang, H., Fan, W., Yu, P., Han, J.: Mining concept-drifting data streams using en-semble classifiers. In: Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–235 (2003)
[14] Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Seventh ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2001), pp. 377–382 (2001)
[15] Karnick, M., Muhlbaier, M.D., Polikar, R.: Incremental Learning in Non-Stationary Environments with Concept Drift Using a Multiple Classifier Based Approach. In: International Conference on Pattern Recognition (ICPR 2008), pp. 1–4 (2008)
[16] Karnick, M., Ahiskali, M., Muhlbaier, M.D., Polikar, R.: Learning concept drift in non-stationary environments using an ensemble of classifiers based approach. In: World Congress on Computational Intelligence, International Joint Conference on Neural Networks, pp. 3455–3462 (2008)
[17] Muhlbaier, M., Polikar, R.: An Ensemble Approach for Incremental Learning in Nonstationery Environments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 490–500. Springer, Heidelberg (2007)
[18] Freund, Y., Schapire, R.E.: Decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)