A NETWORK OF COOPERATIVE LEARNERS FOR DATA–DRIVEN STREAM MINING

Luca Canzian and Mihaela van der Schaar

Department of Electrical Engineering, UCLA, Los Angeles CA 90095, USA.

ABSTRACT

We propose and analyze a distributed learning system to classify data captured from distributed and dynamic data streams. Our scheme consists of multiple distributed learners that are interconnected via an exogenously-determined network. Each learner observes a specific data stream, which is correlated to a common event that needs to be classified, and maintains a set of local classifiers and a weight for each local classifier. We propose a cooperative online learning scheme in which the learners exchange information through the network both to compute an aggregate prediction and to adapt the weights to the dynamic characteristics of the data streams. The information dissemination protocol is designed to minimize the time required to compute the final prediction. We determine an upper bound for the worst-case misclassification probability of our scheme, which depends on the misclassification probability of the best (unknown) static aggregation rule. Importantly, such bound tends to zero if the misclassification probability of the best static aggregation rule tends to zero. When applied to well-known data sets experiencing concept drifts, our scheme exhibits gains ranging from 20% to 70% with respect to state-of-the-art solutions.

Index Terms— Big Data, Stream Mining, Data-Driven Application, Concept Drift, Online Learning, Classification, Networked Learners

1. INTRODUCTION

Recent years have witnessed the proliferation of data-driven applications that exploit the large amount of data captured from distributed data sources. Examples of such applications include surveillance, network monitoring, smart grids, social multimedia, and patient monitoring. However, the effective utilization of such high-volume data also involves significant challenges that are the main concern of this work. First, the statistical properties of the data can evolve in time. Second, the privacy, communication, and sharing costs make it difficult to collect and store all the observed data.

The material is based upon work funded by the US Air Force Research Laboratory (AFRL). Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not reflect the views of AFRL.

The work of Luca Canzian and Mihaela van der Schaar was partially supported by the NSF grant CCF 1218136.

To address these challenges, in this work we propose an online learning scheme [1] in which a set of distributed learners, interconnected via an exogenously-determined network, observe different data streams and exchange pre-processed information to classify a common event and to adapt their configurations to the dynamic characteristics of the data streams. Specifically, each learner maintains a set of local classifiers and a weight for each local classifier, and generates a weighted local prediction about the common event based on the locally observed data. The weighted local predictions are exchanged and combined by each learner using a majority rule to compute the final prediction. After the final prediction has been generated, the learners observe the *label* – the true value of the event to be classified – and exploiting such information they update their local weights adopting a Perceptron learning rule [2].

We determine an upper bound for the worst-case misclassification probability of our scheme which depends on the misclassification probability of the best (unknown) static aggregation rule. Importantly, the misclassification probability of our scheme tends to 0 if the misclassification probability of the best static aggregation rule tends to 0. Since in the presence of *concept drift* [3] (i.e., non-stationary data streams) it is highly unlikely that the misclassification probability of the best static aggregation rule tends to 0, we also extend such an asymptotic result considering assumptions that are more appropriate in the presence of concept-drifting data streams. When applied to well-known data sets experiencing concept drifts, our scheme exhibits gains ranging from 20% to 70% with respect to state-of-the-art solutions.

Much work has been done in the past decade on the development of *online ensemble learning techniques* [4, 5, 6]. An online version of Adaboost [7] is described in [8], and similar proposals are made in [9] and [10]. Weighted majority [11] maintains a collection of given learners and their weights, predicts using a weighted majority rule, and decreases (in a multiplicative manner) the weights of the learners in the pool that disagree with the label whenever the ensemble makes a mistake. Modified versions of weighted majority are proposed in [12] and [13]. [14] proposes a scheme based on two online ensembles, one used for system predictions, the other one used to learn the new concept after a drift is detected.

All the above learning techniques consider a centralized scenario, in which the learners observe the same data and can

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

be centrally retrained. The present study is related to these works and exploits a learning scheme which is similar to that adopted by the weighted majority schemes, but it is focused on a distributed scenario, in which the learners observe different data streams and are connected via an exogenously-determined network, which constrains their communication capabilities. In fact, our learning scheme is proposed and evaluated not only for its prediction accuracy, but also for its capability to generate timely predictions in a distributed environment.

The rest of this paper is organized as follows. Section 2 presents our formalism, framework, and algorithm for distributed online learning. Section 3 characterizes the prediction delay of the proposed system and proves a bound for the misclassification probability of our scheme. Section 4 presents the empirical evaluation of our algorithm on several data sets. Section 5 concludes the paper.

2. THE PROPOSED COOPERATIVE DISTRIBUTED LEARNING SCHEME

We consider a set of L *learners* and we divide time into *slots*. At the beginning of the n-th time slot, each learner i observes a multi-dimensional *instance* or feature vector $\mathbf{x}_i^{(n)} \in \mathcal{X}_i$, which is correlated to a common and unknown *label* $y^{(n)} \in \{-1, 1\}$. The task of each learner is to predict the label $y^{(n)}$.

Each learner i is equipped with a set $\{f_{i,k}^{(n)}\}$ of K_i *local classifiers*. Each local classifier $f_{i,k}^{(n)} : \mathcal{X}_i \to \{-1, 1\}$ generates the *local prediction* $s_{i,k}^{(n)} \triangleq f_{i,k}^{(n)}[\mathbf{x}_i^{(n)}]$ at time slot n. The classifier $f_{i,k}^{(n)}$ can either be a pre-trained static classifier or an incremental classifier that varies with time n [15]. In both cases, we assume that its form is given so that its design is not the focus of this work: our emphasis will be on how to aggregate the local predictions made by the classifiers. We denote by $K \triangleq \sum_i K_i$ the total number of local classifiers in the system. In addition to the set of classifiers, we assume that learner i maintains a set of K_i *weights*, one weight for each of its local classifiers. We denote by $w_{i,k}^{(n)}$ the weight that refers to classifier $f_{i,k}^{(n)}$ in time instant n. We define the *contribution* $C_i^{(n)}$ of learner i to the final prediction in time instant n as

$$C_i^{(n)} \triangleq \sum_{k=1}^{K_i} w_{i,k}^{(n)} s_{i,k}^{(n)}$$

We assume that the learners are connected via an exogenously determined *network* $\mathcal{G}$, which is defined as the set of *links* among pairs of learners. We say that there is a *link* (i, j) between learners i and j if they can communicate directly. The distance $d_{i,j}$ between i and j is defined as the minimum number of links which separates i from j. The diameter $D(\mathcal{G}) \triangleq \max_{i,j} d_{i,j}$ of the network $\mathcal{G}$ is the maximum among all the distances.

We consider a *minimum diameter spanning tree* $\overline{\mathcal{G}}$ of the original network $\mathcal{G}$ [16], i.e., among all the acyclic connected networks obtained by cutting some links from $\mathcal{G}$, $\overline{\mathcal{G}}$ has the smallest diameter. If $D(\overline{\mathcal{G}})$ is even there is a unique learner – which we refer to as *sink learner* – whose distance from the other learners is at maximum $D(\overline{\mathcal{G}})/2$. If $D(\overline{\mathcal{G}})$ is odd there are two linked learners – which we refer to as *sink learners* – whose distance from the other learners is at maximum $(D(\overline{\mathcal{G}}) + 1)/2$. We define the *depth* $\delta_i(\overline{\mathcal{G}})$ of learner i in $\overline{\mathcal{G}}$ as the distance between i and the farther sink learner, we have $\max_i \delta_i(\overline{\mathcal{G}}) = \lceil \frac{D(\mathcal{G})}{2} \rceil$, where $\lceil z \rceil$ denotes the smallest integer equal to or larger than z. If i and j are linked and $\delta_j(\overline{\mathcal{G}}) = \delta_i(\overline{\mathcal{G}}) + 1$ we say that i is the *parent* of j and j is a *child* of i. We denote by $\underline{\mathcal{N}}_i$ the *set of children* of i.

The proposed cooperative distributed learning scheme initializes the weights $w_{i,k}^{(1)} = 0, \forall\, i, k$, and in each time slot n is described through the following five sequential phases:

1) Observation phase. Each learner i observes the data $\mathbf{x}_i^{(n)}$ and computes its contribution $C_i^{(n)}$.

2) Exchange of the aggregate contributions. We divide this phase into $\lceil \frac{D(\mathcal{G})}{2} \rceil$ stages. For each stage $s = 1, \ldots, \lceil \frac{D(\mathcal{G})}{2} \rceil$, each learner i having depth $\delta_i(\overline{\mathcal{G}}) = \lceil \frac{D(\mathcal{G})}{2} \rceil + 1 - s$ sends to its parent the message $m_i^{(n)} = C_i^{(n)} + \sum_{j \in \underline{\mathcal{N}}_i} m_j^{(n)}$, where $m_j^{(n)}$ is the message i received in the preceding stage from its child $j \in \underline{\mathcal{N}}_i$.

3) Computation and exchange of the final prediction. Each sink learner computes the final prediction $\hat{y}^{(n)}$ adopting a *weighted majority rule* [8, 11, 12, 13],

$$\hat{y}^{(n)} = \operatorname{sgn}\left(\mathbf{w}^{(n)} \cdot \mathbf{s}^{(n)}\right) \tag{1}$$

where $\mathbf{w}^{(n)} \triangleq (w_{1,1}^{(n)}, \ldots, w_{1,K_1}^{(n)}, w_{2,1}^{(n)}, \ldots, w_{L,K_L}^{(n)}) \in \Re^K$ and $\mathbf{s}^{(n)} \triangleq (s_{1,1}^{(n)}, \ldots, s_{1,K_1}^{(n)}, s_{2,1}^{(n)}, \ldots, s_{L,K_L}^{(n)}) \in \{-1, 1\}^K$ are the *weight and local prediction vectors*, respectively, $\mathbf{w}^{(n)} \cdot \mathbf{s}^{(n)} \triangleq \sum_{i=1}^{L} C_i^{(n)}$ is the inner product between $\mathbf{w}^{(n)}$ and $\mathbf{s}^{(n)}$, and $\operatorname{sgn}(\cdot)$ is the sign function (we define $\operatorname{sgn}(0) \triangleq 1$). The final prediction $\hat{y}^{(n)}$ is then spread in the network through $D(\mathcal{G}) - \lceil \frac{D(\mathcal{G})}{2} \rceil$ stages in which each learner forwards $\hat{y}^{(n)}$ to its children.

4) Feedback phase. The learners observe the label $y^{(n)}$.

5) Weights update phase. Each learner i compares $\hat{y}^{(n)}$ with $y^{(n)}$ and use this information to update the weights $w_{i,k}^{(n)}$, $k = 1, \ldots, K_i$, of its local classifiers. Specifically, we apply the Perceptron learning rule [2] to the aggregation rule (1), obtaining the following update rule

$$w_i^{(n+1)} = \begin{cases} w_{i,k}^{(n)} & \text{if } \hat{y}^{(n)} = y^{(n)} \\ w_{i,k}^{(n)} + 1 & \text{if } \hat{y}^{(n)} \neq y^{(n)} \text{ and } s_{i,k}^{(n)} = y^{(n)} \\ w_{i,k}^{(n)} - 1 & \text{if } \hat{y}^{(n)} \neq y^{(n)} \text{ and } s_{i,k}^{(n)} \neq y^{(n)} \end{cases} \tag{2}$$

Learner i maintains the same weights if the final prediction agrees with the observed label, $\hat{y}^{(n)} = y^{(n)}$. If disagreement

occurs, then learner i increases by one unit the weights of its classifiers that agree with the label, and decreases by one unit the weights of its classifiers that disagree with the label. Every other learner in the network implements a similar strategy. Notice that $w_{i,k}^{(n)}$ is always an integer number, $\forall\, i,k,n$.

3. PERFORMANCE

In this section, we evaluate the performance of the proposed learning scheme.

In Subsection 3.1 we characterize the *prediction delay* – the time to output the final prediction – and we show that for a large class of networks the proposed information dissemination protocol is *optimal* – it minimizes the prediction delay.

In Subsection 3.2 we analyze the *misclassification probability* – the number of prediction mistakes per instance – and we prove an upper bound for the misclassification probability of our scheme that depends on the misclassification probability of the best (unknown) static weighted majority aggregation rule. Importantly, the resulting bound tends asymptotically to 0 if the misclassification probability of the best static aggregation rule tends to 0.

3.1. Prediction delay

In many stream mining applications it is vital to take *timely decision* based on *timely predictions*. In a distributed environment, the time required to compute the final prediction $\hat{y}^{(n)}$ is constrained by the *prediction delay* $T(\mathcal{G})$, i.e., the time needed to spread the required information in the network.

We consider a positive and monotonically increasing *link delay function* $f : \mathbb{N} \rightarrow \Re^{+}$.[1] We interpret $f(x)$ as the time required to transmit x bits over a link. Alternatively, $f(x)$ can be interpreted as a cost, e.g., the energy required to transmit x bits over a link or the price to pay to access the information of another learner. We denote by x_{boo} and x_{int} the number of bits – included overhead bits – required to transmit a Boolean value and an integer number, respectively.

Proposition 1 characterizes the prediction delay $T^P(\mathcal{G})$ for the proposed scheme.

Proposition 1 *The prediction delay $T^P(\mathcal{G})$ of the proposed scheme is*

$$T^P(\mathcal{G}) = \left\lceil \frac{D(\overline{\mathcal{G}})}{2} \right\rceil f(x_{int}) + \left(D(\overline{\mathcal{G}}) - \left\lceil \frac{D(\overline{\mathcal{G}})}{2} \right\rceil \right) f(x_{boo})$$

Proposition 1 shows that the prediction delay depends only on the diameter of the minimum diameter spanning tree network $\overline{\mathcal{G}}$ and hence increases slowly in the number of learners L.[2] Notice the proposed dissemination protocol is not simply a matter of networking, it exploits the particular structure of the proposed aggregation and learning rules, (1) and (2), to efficiently combine data at intermediate learners.

Proposition 2 shows that the considered information dissemination protocol allows to minimize the prediction delay for the class $\mathcal{C}_{DP}$ of diameter-preserving spanning tree networks, i.e., for each network $\mathcal{G}$ such that $D(\mathcal{G}) = D(\overline{\mathcal{G}})$. Acyclic networks are an example of such networks.

Proposition 2 *Let $\tilde{P}$ an information dissemination protocol such that all learners are able to compute the final prediction (2). If $\mathcal{G} \in \mathcal{C}_{DP}$ then $T^{\tilde{P}}(\mathcal{G}) \geq T^P(\mathcal{G})$*

3.2. Misclassification probability

We denote by $q^{(n)}(\mathbf{x}_1^{(n)}, \ldots, \mathbf{x}_L^{(n)}, y^{(n)})$ the probability that the learners observe the instances $\mathbf{x}_1^{(n)}, \ldots, \mathbf{x}_K^{(n)}$ and the label is $y^{(n)}$. As in [3], we refer to a particular probability distribution q_c as a *concept*, we say that the *concept is stable* if $q^{(n)}$ is an independent and identically distributed process with $q^{(n)} = q_c$, $\forall\, n$, whereas we say that at time instant n there is a *concept drift* if $q^{(n+1)} \neq q^{(n)}$.

Given the sequence of N instances and labels

$$\mathbf{D}_N \triangleq \left(\mathbf{x}_1^{(n)}, \ldots, \mathbf{x}_K^{(n)}, y^{(n)} \right)_{n=1,\ldots,N} ,$$

we denote by $P^O(\mathbf{D}_N)$ the misclassification probability of a learner that combines the local predictions of all the classifiers in the system with the *optimal static weight vector* $\mathbf{w}^O$ that minimizes its number of mistakes,

$$P^O(\mathbf{D}_N) \triangleq \min_{\mathbf{w}^O} \frac{1}{N} \sum_{n=1}^{N} I\{\mathrm{sgn}\left(\mathbf{w}^O \cdot \mathbf{s}^{(n)}\right) \neq y^{(n)}\}$$

We remark that the computation and adoption of $\mathbf{w}^O$ would require to know in advance, at the beginning of the first time slot, the sequences of local predictions $\mathbf{s}^{(n)}$ and labels $y^{(n)}$, for every time slot $n = 1, \ldots, N$.

Moreover, we denote by $P(\mathbf{D}_N)$ the misclassification probability of a learner if all the learners adopt the proposed scheme,

$$P(\mathbf{D}_N) \triangleq \frac{1}{N} \sum_{n=1}^{N} I\{\mathrm{sgn}\left(\mathbf{w}^{(n)} \cdot \mathbf{s}^{(n)}\right) \neq y^{(n)}\}$$

where $w_{i,k}^{(1)} = 0$ and $w_{i,k}^{(n)}$ evolves according to (2), for each learner $i = 1, \ldots, L$ and classifier $k = 1, \ldots K_i$.

Theorem 1 determines an upper bound for $P(\mathbf{D}_N)$ in terms of the unknown benchmark $P^O(\mathbf{D}_N)$.

Theorem 1 *For every sequence of labeled instances $\mathbf{D}_N$, the misclassification probability $P(\mathbf{D}_N)$ is upper bounded by*

$$\mathbf{B}(\mathbf{D}_N) \triangleq (1+\sqrt{K})P^O(\mathbf{D}_N) + 2\sqrt{\frac{(K+\sqrt{K})P^O(\mathbf{D}_N)}{N} + \frac{K}{N}}$$

[1] The proposed information dissemination protocol can be generalized considering link-dependent delay functions $f_{i,j}$, for each link (i,j).

[2] [17] proves that the diameter of a random acyclic network grows as $O(\sqrt{L})$.

Importantly, notice that the bound $\mathbf{B}(\mathbf{D}_N)$ is valid for any time horizon N and for any sequence of labeled instances $\mathbf{D}_N$. As a particular case, if the time horizon tends to infinity and $P^O(\mathbf{D}_N)$ tends to 0, the misclassification probability of our scheme tends to 0 as well.

Corollary 1 *If* $\lim_{N\to+\infty} P^O(\mathbf{D}_N) = 0$*, then*

$$\lim_{N\to+\infty} P(\mathbf{D}_N) = 0$$

Corollary 1 can be useful if the concept is stable, but it is not useful if there is concept drift because the accuracies of the classifiers and, consequently, the optimal weight vector $\mathbf{w}^O$ can change consistently from one concept to another. Now we generalize the result of Corollary 1 considering an assumption that is more realistic if there are concept drifts.

We denote by $\mathbf{D}_{N_c}$ a sequence of N_c instances and labels generated by the concept q_c. We say that the concept q_c is *learnable* if, $\forall\, \mathbf{D}_{N_c}$,

$$\lim_{N_c\to+\infty} \min_{\mathbf{w}_c^O} \frac{1}{N_c} \sum_{n=1}^{N_c} I\{\text{sgn}\left(\mathbf{w}_c^O \cdot \mathbf{s}^{(n)}\right) \neq y^{(n)}\} = 0$$

That is, the concept $q_c^{(n)}$ is learnable if there exists a static weight vector $\mathbf{w}_c^O$ whose asymptotic misclassification probability, over the instances and labels generated by that concept, tends to 0.

Theorem 2 *If* $\mathbf{D}_N$*, for* $N \to +\infty$*, is generated by a finite number of learnable concepts and a finite number of concept drifts occurred, then*

$$\lim_{N\to+\infty} P(\mathbf{D}_N) \to 0$$

We remark that Corollary 1 requires the existence of a *unique weight vector*, $\mathbf{w}^O$, whose misclassification probability over the labeled instances generated by *all* concepts converges to 0; whereas Theorem 2 requires the existence of *one weight vector for concept*, $\mathbf{w}_c^O$, whose misclassification probability over the labeled instances generated by concept q_c converges to 0.

4. SIMULATIONS

In this section we evaluate empirically the performance of our scheme and compare it with the ensemble learning techniques listed in Table 1. For each scheme we adopt the parameters used in the corresponding paper. We consider three data sets widely used by the literature dealing with concept drift, that refer to real-world problems: in **R1** the task is to detect a network attack [10, 14, 18, 19]; in **R2** the task is to detect if the price of the energy will increase or decrease [14, 20, 21]; and in **R3** the task is to classify the forest cover type of a particular area [10, 20, 22, 19]. For a detailed description of the considered schemes and data sets, we refer the reader to the cited references.

Table 1. The considered schemes and their percentages of misclassifications in the data sets **R1**-**R3**

Name and Reference	Performance		
	R1	**R2**	**R3**
Average Majority, [18]	3.07	41.8	29.5
Adaboost, [7]	5.25	41.1	57.5
Weighted Majority (WM), [11]	0.29	22.9	14.1
Blum's variant of WM, [12]	1.64	37.3	22.6
TrackExp, [13]	0.52	23.1	14.8
Our scheme	**0.23**	**14.4**	**4.1**

For each data set we consider a set of 8 logistic regression classifiers [23]. The training and testing procedures are as follows. From the whole data set we select 8 training data sets, each of them consisting of Z sequential records. Z is equal to $5,000$ for the data sets **R1** and **R3**, and $2,000$ for **R2**. Then we take other sequential records ($20,000$ for **R1** and **R3**, and $8,000$ for **R2**) to generate a set in which the local classifiers are tested, and the results are used to train Adaboost. Finally, we select other sequential records ($20,000$ for **R1** and **R3**, $21,000$ for **R2**) to generate the testing set which is used to run the simulations and test all the considered schemes.

Table 1 reports the final misclassification probability in percentages (i.e., multiplied by 100) obtained for each data set for the considered schemes. The schemes that update their models (WM, Blum's variant of WM, TrackExp, and our scheme) outperform the static schemes (average majority and Adaboost); in fact, the latter do not adapt to changes in concept. Importantly, in all the considered data sets our scheme outperforms the other schemes and exhibits performance gains ranging from 20% (**R1**) to 70% (**R3**) with respect to WM, the second best scheme.

5. CONCLUSION

We proposed a distributed online ensemble learning scheme to classify data captured and pre-processed by multiple distributed local learners. We designed an efficient information dissemination protocol to spread the required information in the network. We rigorously determined a bound for the worst-case misclassification probability of our scheme which depends on the misclassification probability of the best static aggregation rule. Importantly, this bound tends asymptotically to 0 if the misclassification probability of the best static aggregation rule tends to 0. Simulation results show that, when applied to well-known data sets experiencing concept drifts, our scheme exhibits performance gains ranging from 20% to 70% with respect to state-of-the-art solutions.

6. REFERENCES

[1] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, 2004.

[2] F. Rosenblatt, "The perceptron: a perceiving and recognizing automaton," Tech. Rep., Cornell Aeronautical Laboratory, 1957.

[3] I. Zliobaite, "Learning under concept drift: an overview," Tech. Rep., Vilnius University, Faculty of Mathematics and Informatics, 2010.

[4] Z. Haipeng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: Models, limits, and algorithms," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 386–398, 2011.

[5] T. Shinozaki, Y. Kubota, and S. Furui, "Unsupervised acoustic model adaptation based on ensemble methods," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 6, pp. 1007–1015, 2010.

[6] V. H. Nascimento and A. H. Sayed, "On the learning mechanism of adaptive filters," *IEEE Trans. Signal Process.*, vol. 48, no. 6, pp. 1609–1625, 2000.

[7] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. ICML*, 1996, pp. 148–156.

[8] W. Fan, S. J. Stolfo, and J. Zhang, "The application of AdaBoost for distributed, scalable and on-line learning," in *Proc. ACM SIGKDD*, 1999, pp. 362–366.

[9] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. ACM SIGKDD*, 2003, pp. 226–235.

[10] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Integrating novel class detection with classification for concept-drifting data streams," in *Proc. ECML PKDD*, 2009, pp. 79–94.

[11] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[12] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Mach. Learn.*, vol. 26, no. 1, pp. 5–23, Jan. 1997.

[13] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, Aug. 1998.

[14] L. L. Minku and Y. Xin, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, 2012.

[15] D. Shutin, S. R. Kulkarni, and H. V. Poor, "Incremental reformulated automatic relevance determination," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4977–4981, 2012.

[16] M. Bui, F. Butelle, and C. Lavault, "A distributed algorithm for constructing a minimum diameter spanning tree," *J. Parallel Distrib. Comput.*, vol. 64, no. 5, pp. 571–577, May 2004.

[17] G. Szekeres, "Distribution of labelled trees by diameter," *Lecture Notes in Math.*, vol. 1036, pp. 392–397, 1983.

[18] J. Gao, W. Fan, and J. Han, "On appropriate assumptions to mine data streams: Analysis and practice," in *Proc. IEEE ICDM*, 2007, pp. 143–152.

[19] K. Bache and M. Lichman, "UCI machine learning repository," Tech. Rep., University of California, Irvine, School of Information and Computer Sciences, 2013.

[20] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, "Fast perceptron decision tree learning from evolving data streams," in *Proc. PAKDD*, 2010, pp. 299–310.

[21] M. Harries, "SPLICE–2 comparative evaluation: Electricity pricing," Tech. Rep., University of New South Wales, School of Computer Science and Engineering, 1999.

[22] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proc. ACM SIGKDD*, 2001, pp. 359–364.

[23] D. S. Rosario, "Highly effective logistic regression model for signal (anomaly) detection," in *Proc. IEEE ICASSP*, 2004, vol. 5, pp. 817–820.