Context-Adaptive Big Data Stream Mining

Cem Tekin, Luca Canzian, Mihaela van der Schaar

Electrical Engineering Department, University of California, Los Angeles
Email: cmtkn@ucla.edu, luca.canzian@gmail.com, mihaela@ee.ucla.edu

***Abstract*—Emerging stream mining applications require classification of large data streams generated by single or multiple heterogeneous sources. Different classifiers can be used to produce predictions. However, in many practical scenarios the distribution over data and labels (and hence the accuracies of the classifiers) may be unknown a priori and may change in unpredictable ways over time. We consider data streams that are characterized by their context information which can be used as meta-data to choose which classifier should be used to make a specific prediction. Since the context information can be high dimensional, learning the best classifiers to make predictions using contexts suffers from the curse of dimensionality. In this paper, we propose a context-adaptive learning algorithm which learns online what is the best context, learner, and classifier to use to process a data stream. Then, we theoretically bound the regret of the proposed algorithm and show that its time order is independent of the dimension of the context space. Our numerical results illustrate that our algorithm outperforms most prior online learning algorithms, for which such online performance bounds have not been proven.**

***Index Terms*—Stream mining, context-adaptive learning, distributed multi-user learning, contextual bandits.**

I. Introduction

A plethora of Big Data applications including network monitoring [1], surveillance [2] and health monitoring [3] (see Fig. 1), are emerging which require online classification of large data sets collected from single or distributed sensors, monitors, multimedia sources, etc. The data streams collected by such sources are heterogeneous and dynamically evolving over time in unknown and unpredictable ways. Hence, mining these data streams online, at run-time, is known to be a very challenging problem [4], [5]. In this paper, we tackle these online stream mining challenges by exploiting the automatically generated meta-data, referred to as "contexts", which is gathered or associated to the data streams in the process of capturing or pre-processing them. Contexts can represent any side-information related to the input data stream such as the location at which the data was captured, and/or data type information (e.g., data features/characteristics/modality). We assume that each data stream is processed by a decision maker/learner, which upon receiving the data and associated contexts takes a classification action (i.e., calls a local classifier or another learner), which will return a prediction.

Classifiers can be anything ranging from separating hyperplanes, naive Bayesian classifiers, random trees, etc., whose expected accuracies are unknown a priori to the decision maker/learner. The focus of this paper is to determine how to learn online, based on the data and the labels (feedback)

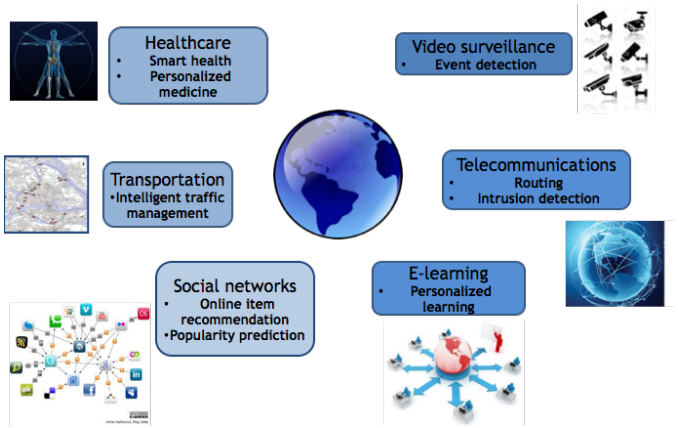

Fig. 1. Various applications of Big Data stream mining.

received in the past, which type of context is the most relevant given a vector of available contexts at each time instance, and to choose the *marginally* best classifier (have the highest accuracy given the value of a particular type of context but not the entire context vector) according to its estimated performance for that particular context. The goal of each learner is to maximize its long term expected total reward, which is the expected number of correct labels minus the costs of classification. In this paper the cost is a generic term that can represent any known cost such as processing cost, delay cost, communication cost, etc. Similarly, data is used as a generic term. It can represent files of several Megabytes size, chunks of streaming media packets or contents of web pages.

If multiple learners are available in the system to capture and process various data streams (possibly captured at different locations), they can cooperate with each other to process the streams. Each learner can then process (make a prediction on) the incoming data in two different ways: either it can exploit one of its own classifiers to make a prediction or it can forward its input stream to another learner (possibly by incurring some cost) to have it labeled. A learner learns the accuracies of its own classifiers or other learners in an online way, by comparing the result of the predictions with the true label of its input stream which is revealed at the end of each time slot. We consider cooperative learners which classify other's data when requested, such that the individual utility (expected total reward) of each learner is maximized.

Each learner faces a sequential decision making problem of

what classification action to take based on its current context vector and its history of observations and decisions. Therefore, our focus in this paper is how to learn the best classifiers from a given set of classifiers rather than designing the classifiers. Since multi-armed bandit (or simply, bandit) [6]–[9] based approaches are proven to be very effective for sequential decision making problems, in this paper we propose a novel bandit approach which adaptively learns to make decisions based on the values of different types of contexts.

The context information is usually correlated with the data and the label, and can significantly improve the classification performance if used smartly. However, when the number of types of contexts associated with the data is large (i.e., dimension of the context vector is large), learning the best classification action according to the entire set of contexts suffers from the curse of dimensionality. In this paper we propose a new approach in which the marginally best classification action at each time instance is learned based on the best type of context in the current set of contexts. We will show that this solution does not suffer from the curse of dimensionality.

The remainder of the paper is organized as follows. In Section II, we describe the related work. In Section III, we formalize the problem and define the regret. We propose a distributed online learning algorithm which learns to exploit the marginally best classification action over time in Section IV. Numerical results are given in Section V. Finally, the concluding remarks are given in Section VI.

II. Related Work

Most of the prior work in stream mining is focused on learning from the unique data stream characteristics [10]–[15]. In this paper, we take a different approach: instead of focusing on the characteristics of a specific data stream, we focus on the characteristics of data streams having the same context information. Importantly, we focus on learning what type of context information should be taken into account when choosing a classifier to make a prediction. Some context can reveal a lot of information about the best action to take, while some other context may be irrelevant. We provide a detailed comparison to our work in Table I.

Other than distributed data mining, our learning framework can be applied to any problem that can be formulated as a decentralized contextual bandit problem [16]. Contextual bandits have been studied before in [8], [9], [17], [18] and other works in a single agent setting. However our work is very different from these because (i) we consider decentralized agents who can learn to cooperate with each other, (ii) the set of actions and realization of data and context arrivals to the agents can be very different for each agent, (iii) instead of learning to take the best action considering the entire D-dimensional context vector, an agent learns to take the marginally best action by independently considering each D types of contexts.

Our prior work has shown that the performance of existing learning algorithms usually depends on the dimension of the context space [16]. When the dimension of the context space is large, the convergence rate of these algorithms to the optimal average reward becomes very slow. However, the convergence speed of the algorithms we propose in this paper is independent of this dimension.

III. Problem Formulation

The system model is shown in Fig. 2 and 3. There are M learners which are indexed by the set $\mathcal{M} := \{1, 2, \ldots, M\}$. The set of classifiers learner i has is $\mathcal{F}_i$. The set of all classifiers is $\mathcal{F} = \cup_{i\in\mathcal{M}}\mathcal{F}_i$. Let $\mathcal{M}_{-i} := \mathcal{M} - \{i\}$ be the set of learners learner i can choose from to send its data for classification. The action (arm) set[1] of learner i is $\mathcal{K}_i := \mathcal{F}_i \cup \mathcal{M}_{-i}$. Throughout the paper we use index f to denote an element of $\mathcal{F}$, j to denote learners in $\mathcal{M}_{-i}$, and k to denote an element of $\mathcal{K}_i$.

These learners work in a discrete time setting $t = 1, 2, \ldots, T$, where the following events happen sequentially, in each time slot: (i) data $s_i(t) \in \mathcal{S}$ with a specific D-dimensional context vector $\boldsymbol{x}_i(t) = (x_i^1(t), \ldots, x_i^D(t))$ arrives to each learner $i \in \mathcal{M}$, where $\mathcal{S}$ is the data set, $x_i^d(t) \in \mathcal{X}_d$ for $d \in \mathcal{D} := \{1, \ldots, D\}$ and $\mathcal{X}_d$ is the set of type-d contexts, and $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_D$ is the context space,[2] (ii) each learner i chooses one of its own classifiers or another learner to send its data and context, and produces a label $\hat{y}_i(t) \in \mathcal{Y}$ based on the prediction of its own classifier or the learner to which it sent its data and context, where $\mathcal{Y}$ is the set of possible labels, (iii) the truth (true label) $y_i(t) \in \mathcal{Y}$ is revealed, perhaps by events or by a supervisor, only to the learner i where the data arrived, (iv) the learner where the data arrived passes the true label to the learner it had chosen to classify its data, if there is such a learner.

A. Data, label, context and classifier accuracies

At each time slot $s_i(t)$, $y_i(t)$ and $\boldsymbol{x}_i(t)$ are drawn from an (unknown) joint distribution J over $\mathcal{S} \times \mathcal{Y} \times \mathcal{X}$ independently from other time slots for each learner $i \in \mathcal{M}$. We do not require this draw to be independent among the learners. Since context vector $\boldsymbol{x}_i(t)$ is revealed to learner i at the beginning of time t, depending on J, there exists a conditional distribution $G_{\boldsymbol{x}_i(t)}$ over $\mathcal{S} \times \mathcal{Y}$. Similarly, depending on J, there is a marginal distribution H over $\mathcal{X}$ from which contexts are drawn. Given context vector $\boldsymbol{x}$, let

$$\begin{aligned}\pi_f(\boldsymbol{x}) &:= \mathrm{E}[\mathrm{I}(f(s_i(t)) = y_i(t))] \\ &= \int_{(s,y)\in\mathcal{S}\times\mathcal{Y}} \mathrm{I}(f(s_i(t)) = y_i(t)) dG_{\boldsymbol{x}}(s, y)\end{aligned}$$

be the *joint accuracy* (or simply, accuracy) of classifier $f \in \mathcal{F}$, where $f(s_i(t))$ is the prediction of classifier f on the data, $\mathrm{I}(\cdot)$ is the indicator function which is equal to 1 if the statement inside is true and 0 otherwise, and the expectation $\mathrm{E}[\cdot]$ is taken with respect to distribution $G_{\boldsymbol{x}}$. Let

[1] In sequential online learning literature [6], [7], an action is also called an arm (or an alternative).

[2] In our analysis, we will assume that $\mathcal{X}_d = [0, 1]$ for all $d \in \mathcal{D}$. However, our algorithms will work and our results will hold even when the context space is discrete given that it is bounded.

	[12], [19]–[22]	[15], [23]	[13]	[16]	This work
Aggregation	non-cooperative	cooperative	cooperative	no	no
Message exchange	none	data	training residual	data and label (adaptively)	data and label (adaptively)
Learning approach	offline/online	offline	offline	Non-Bayesian online	Non-Bayesian online
Learning from other's contexts	N/A	no	no	yes	yes
Using other's classifiers	no	all	all	sometimes-adaptively	sometimes-adaptively
Data partition	horizontal	horizontal	vertical	both	both
Bound on regret	no	no	no	yes - context dependent	yes - context independent
Context adaptive	no	no	no	no	yes

TABLE I
COMPARISON WITH RELATED WORK IN DISTRIBUTED DATA MINING.

$\boldsymbol{x}^{-d} := (x^1, \ldots, x^{d-1}, x^{d+1}, \ldots, x^D)$ and $((\boldsymbol{x}')^{-d}, x^d) = (x'^1, \ldots, x'^{d-1}, x^d, x'^{d+1}, \ldots, x'^D)$. Then, the *marginal accuracy* of f based on type-d context is defined as

$$\pi_f^d(x^d) := \int_{(\boldsymbol{x}')^{-d}} \pi_f((\boldsymbol{x}')^{-d}, x^d) dH((\boldsymbol{x}'^{-d}), x^d).$$

We say that the problem has *the similarity property* when each classifier has similar marginal accuracies for similar contexts.

Definition 1: **Similarity Property.** If the joint distribution J over $\mathcal{S} \times \mathcal{Y} \times \mathcal{X}$ is such that for each $f \in \mathcal{F}$ and $d \in \mathcal{D}$, there exists a minimum $\alpha > 0$ and a minimum $L > 0$, such that for all $x^d, (x')^d \in \mathcal{X}_d$, we have

$$|\pi_f^d(x^d) - \pi_f^d((x')^d)| \leq L|x^d - (x')^d|^\alpha,$$

then we call J *a distribution with similarity*.

Although, our model assumes a continuous context space, our algorithms will also work when the context space is discrete. Note that Definition 1 does not require the context space to be continuous. We assume that α is known by the learners, while L does not need to be known. However, our algorithms can be combined with estimation methods for α.

B. Unknowns, actions and rewards

In our problem, the unknowns for learner i are (i) $\mathcal{F}_j$, $j \in \mathcal{M}_{-i}$, (ii) J, H, $G_{\boldsymbol{x}}$, $\boldsymbol{x} \in \mathcal{X}$, (iii) $\pi_f(\boldsymbol{x})$, $f \in \mathcal{F}_i$, $\boldsymbol{x} \in \mathcal{X}$, (iv) $\pi_f^d(x^d)$, $f \in \mathcal{F}_i$, $x^d \in \mathcal{X}_d$, $d \in 1, \ldots, D$.

On the other hand, learner i knows (i) the functions in $\mathcal{F}_i$ and costs of calling them,[3] (ii) the set of other learners $\mathcal{M}_{-i}$ and costs of calling them, (iii) and an upper bound on the number of classifiers that each learner has, i.e., $F_{\max} \geq |\mathcal{F}_j|$,[4] for all $j \in \mathcal{M}_{-i}$.

At each time slot t, learner i can either invoke one of its classifiers or forward the data to another learner to have it labeled. We assume that for learner i, calling each classifier $f \in \mathcal{F}_i$ incurs a cost $c_f^i \geq 0$. A learner i can also send its data to another learner in $\mathcal{M}_{-i}$ in order to have it labeled. Because of the communication cost and the delay caused by processing at the recipient, we assume that whenever the data is sent to another learner $j \in \mathcal{M}_{-i}$ a cost of c_j^i is incurred by learner i.[5] Since the costs are bounded, without loss of generality we assume that costs are normalized, i.e., $c_k^i \in [0, 1]$ for all $k \in \mathcal{K}_i$.[6] The learners are cooperative which implies that learner $j \in \mathcal{M}_{-i}$ will return a prediction to i when called by i using its classifier with the highest estimated accuracy for i's context vector. Similarly, when called by $j \in \mathcal{M}_{-i}$, learner i will return a label to j. In our theoretical analysis we do not consider the effect of this on i's learning rate; however, since our results hold for the case when other learners are not helping i to learn about its own classifiers, they will also hold when other learners help i to learn about its own classifiers.

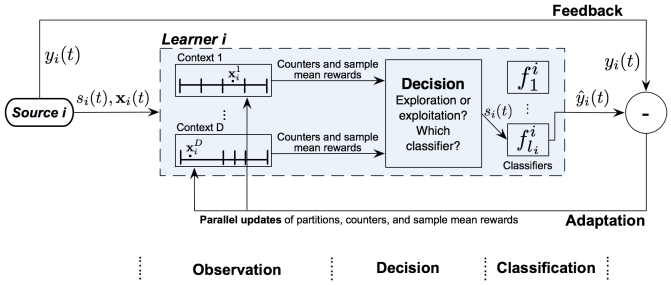

Fig. 2. Operation of learner i during a time slot when it chooses one of its own classifiers.

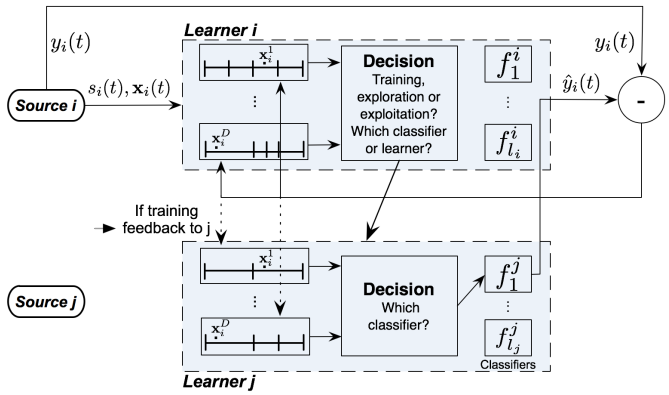

Fig. 3. Operation of learner i during a time slot when it chooses learner j.

[3]Alternatively, we can assume that the costs are random variables with bounded support whose distribution is unknown. In this case, the learners will not learn the accuracy but they will learn accuracy minus cost.

[4]For a set A, let $|A|$ denote the cardinality of that set.

[5]The cost for learner i does not depend on the cost of the classifier chosen by learner j. Since the learners are cooperative, j will obey the rules of the proposed algorithm when choosing a classifier to label i's data. We assume that when called by i, j will select a classifier from $\mathcal{F}_j$, but not forward i's data to another learner.

[6]If there is a classifier f such that it is both in $\mathcal{F}_i$ and $\mathcal{F}_j$, we assume that the cost of calling $f \in \mathcal{F}_i$ is smaller than the cost of calling learner j for learner i.

We assume that each classifier produces a binary label,[7] thus $\mathcal{Y} = \{0, 1\}$. For a learner $j \in \mathcal{M}_{-i}$ its accuracy for a type-d context x^d is equal to the accuracy of its best classifier, i.e., $\pi_j^d(x^d) := \max_{k \in \mathcal{F}_j} \pi_k^d(x^d)$. The goal of each learner i is to maximize its total expected reward. This corresponds to minimizing the regret with respect to the benchmark solution which we will define in the next subsection.

C. Classification with Complete Information

Our benchmark when evaluating the performance of the learning algorithms is the solution which selects the arm in $\mathcal{K}_i$ with the highest marginal accuracy minus cost (i.e., reward) given the context vector $\boldsymbol{x}_i(t)$ at time t. Specifically, the solution we compare against is given by

$$k_i^*(\boldsymbol{x}) := \arg\max_{k \in \mathcal{K}_i} \left(\max_{x^d \in \boldsymbol{x}} \pi_k^d(x^d) - c_k^i \right), \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

Since calculating $k_i^*(\boldsymbol{x})$ requires knowledge of marginal classifier accuracies only, we call $k_i^*(\boldsymbol{x})$ *the marginally best classification action* given context $\boldsymbol{x}$. Knowing this means that learner i knows the classifier in $\mathcal{F}$ that yields the highest accuracy for each $x^d \in \mathcal{X}_d$, $d \in \mathcal{D}$. In general we have $k_i^*(\boldsymbol{x}) \neq \arg\max_{k \in \mathcal{K}_i} \pi_k(\boldsymbol{x}) - c_k^i$. But for special cases, such as when the distribution of data only depends on the value of a single type of context (but this type is unknown), the two are equivalent.

D. The Regret of Learning

Simply, the regret is the loss incurred due to the unknown system dynamics. Regret of a learning algorithm which selects an action $a_i(t) \in \mathcal{K}_i$ at time t for learner i based on its context vector $\boldsymbol{x}_i(t)$ and the past observations is defined as

$$R_i(T) := \sum_{t=1}^{T} \left(\pi_{k_i^*(\boldsymbol{x}_i(t))}(\boldsymbol{x}_i(t)) - c^i_{k_i^*(\boldsymbol{x}_i(t))} \right) - \mathrm{E}\left[\sum_{t=1}^{T} (\mathrm{I}(\hat{y}_i(t) = y_i(t)) - c^i_{a_i(t)}) \right],$$

where $\hat{y}_i(t)$ denotes the prediction of the action $a_i(t)$ selected by learner i at time t, $y_i(t)$ denotes the true label of the data stream that arrived to learner i in time slot t, and the expectation is taken with respect to the randomness of the prediction. Regret gives the convergence rate of the total expected reward of the learning algorithm to the value of the benchmark solution $k_i^*(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$. Any algorithm whose regret is sublinear, i.e., $R_i(T) = O(T^\gamma)$ such that $\gamma < 1$, will converge to the benchmark solution in terms of the average reward.

IV. Adaptively Learning the Contexts to Exploit

In this section we propose an online learning algorithm that achieves sublinear in time regret when the condition in Definition 1 holds. We name our algorithm *Adaptive Contexts and Adaptive Partitions* (ACAP).

A. The ACAP algorithm

The basic idea behind ACAP is to adaptively divide the context space into finer and finer regions over time such that regions of the context space with a large number of arrivals are trained and explored more accurately than regions of the context space with small number of arrivals, and then only use the observations in those sets when estimating the rewards of arms in $\mathcal{K}_i$ for contexts that lie in those sets. At each time slot, ACAP chooses an arm adaptively based on the estimated marginal accuracies of the arms given the context vector. At time t, we call the type d context for which the selected arm's marginal accuracy is maximized as the *main context* of that time.

For each type-d context, ACAP starts with a single hypercube which is the entire context space $\mathcal{X}_d$, then divides the space into finer regions and explores them as more contexts arrive. In this way, the algorithm focuses on parts of the space in which there is large number of context arrivals, and does this independently for each type of context. The learning algorithm for learner i should zoom into the regions of space with large number of context arrivals, but it should also persuade other learners to *zoom* to the regions of the space where learner i has a large number of context arrivals. Here zooming means using past observations from a smaller region of context space to estimate the rewards of actions for a context. The pseudocode of ACAP is given in Fig. 4, and the initialization, training, exploration and exploitation modules are given in Fig. 5 and Fig. 6.

For each type-d context, we call an interval $(a2^{-l}, (a+1)2^{-l}] \subset [0, 1]$ a level l hypercube for $a = 1, \ldots, 2^l - 1$,[8] where l is an integer. Let $\mathcal{P}_l^d$ be the partition of type-d context space $[0, 1]$ generated by level l hypercubes. Clearly, $|\mathcal{P}_l^d| = 2^l$. Let $\mathcal{P}^d := \cup_{l=0}^{\infty} \mathcal{P}_l^d$ denote the set of all possible hypercubes. Note that $\mathcal{P}_0^d$ contains only a single hypercube which is $\mathcal{X}_d$ itself. At each time slot, ACAP keeps for learner i a set of mutually exclusive hypercubes that cover the context space of each type $d \in \mathcal{D}$ context. We call these hypercubes *active* hypercubes, and denote the set of active hypercubes for type-d context at time t by $\mathcal{A}_i^d(t)$. Let $\mathcal{A}_i(t) := (\mathcal{A}_i^1(t), \ldots, \mathcal{A}_i^D(t))$. Clearly, we have $\cup_{C \in \mathcal{A}_i^d(t)} C = \mathcal{X}_d$. Denote the active hypercube that contains $x_i^d(t)$ by $C_i^d(t)$. Let $\boldsymbol{C}_i(t) := (C_i^1(t), \ldots, C_i^D(t))$ be the set of active hypercubes that contains $\boldsymbol{x}_i(t)$. The arm chosen by learner i at time t only depends on the actions taken on previous context observations which are in $C_i^d(t)$ for some $d \in \mathcal{D}$. The number of such actions and observations can be much larger than the number of previous actions and observations in $\boldsymbol{C}_i(t)$. This is because in order for an observation to be in $\boldsymbol{C}_i(t)$, it should be in all $C_i^d(t)$, $d \in \mathcal{D}$. Let $N_C^{i,d}(t)$ be the number of times type-d contexts have arrived to hypercube C of learner i from the activation of C till time t. Once activated, a level l

[7] In general we can assume that labels belong to $\mathbb{R}$ and define the classification error as the mean squared error or some other metric. Our results can be adapted to this case as well.

[8] The first level l hypercube is defined as $[0, 2^{-l}]$.

Adaptive Contexts and Adaptive Partitions Algorithm (for learner i):

Input: $D_1(t)$, $D_2(t)$, $D_3(t)$, p, A
Initialization: $\mathcal{A}_i^d = \{[0,1]\}$, $d \in \mathcal{D}$. $\mathcal{A}_i = \mathcal{A}_i^1 \times \ldots \times \mathcal{A}_i^D$. Run **Initialize**($\mathcal{A}_i$)
Notation: $\bar{\boldsymbol{r}}_k^i = (\bar{r}_{k,C^d(t)}^{i,d})_{d\in\mathcal{D}}$,
$\bar{\boldsymbol{r}}^i = (\bar{\boldsymbol{r}}_k^i)_{k\in\mathcal{K}_i}$,
l_C: level of hypercube C,
$\boldsymbol{N}_k^i = (N_{k,C^d(t)}^{i,d})_{d\in\mathcal{D}}$, $k \in \mathcal{K}_i$,
$\boldsymbol{N}^i = (\boldsymbol{N}_k^i)_{k\in\mathcal{K}_i}$.
while $t \geq 1$ **do**
  if $\exists d \in \mathcal{D}$ and $\exists k \in \mathcal{F}_i$ such that $N_{k,C^d(t)}^{i,d} \leq D_1(t)$ **then**
    Run **Explore**(t, k, d, $N_{k,C^d(t)}^{i,d}$, $\bar{r}_{k,C^d(t)}^{i,d}$)
  else if $\exists d \in \mathcal{D}$ and $\exists k \in \mathcal{M}_{-i}$ such that $N_{1,k,C^d(t)}^{i,d} \leq D_2(t)$ **then**
    Run **Train**(t, k, d, $N_{1,k,C^d(t)}^{i,d}$)
  else if $\exists d \in \mathcal{D}$ and $\exists k \in \mathcal{M}_{-i}$ such that $N_{k,C^d(t)}^{i,d} \leq D_3(t)$ **then**
    Run **Explore**(t, k, d, $N_{k,C^d(t)}^{i,d}$, $\bar{r}_{k,C^d(t)}^{i,d}$)
  else
    Run **Exploit**(t, $\boldsymbol{N}^i$, $\bar{\boldsymbol{r}}^i$, $\mathcal{K}_i$)
  end if
  $N_{C^d(t)}^{i,d} = N_{C^d(t)}^{i,d} + 1$
  for $d \in \mathcal{D}$ **do**
    if $N_{C^d(t)}^{i,d} \geq A2^{pl_{C^d(t)}}$ **then**
      Create 2 level $l_{C_d(t)} + 1$ child hypercubes denoted by $\mathcal{A}_{C^d(t)}$
      Run **Initialize**($\mathcal{A}_{C^d(t)}$)
      $\mathcal{A}_i = \mathcal{A}_i \cup \mathcal{A}_{C^d(t)} - C^d(t)$
    end if
  end for
  $t = t + 1$
end while

Fig. 4. Pseudocode of the ACAP algorithm.

Initialize($\mathcal{A}$):

for $C \in \mathcal{A}$ **do**
  Set $N_C^{i,d} = 0$, $N_{k,C}^{i,d} = 0$, $\bar{r}_{k,C}^{i,d} = 0$ for $k \in \mathcal{K}_i$, $N_{1,k,C}^{i,d} = 0$ for $k \in \mathcal{M}_{-i}$.
end for

Fig. 5. Pseudocode of the initialization module.

hypercube C will stay active until the first time t such that $N_C^{i,d}(t) \geq A2^{pl}$, where $p > 0$ and $A > 0$ are parameters of ACAP. After that, ACAP will divide C into 2 level $l+1$ hypercubes.

For each action (classifier) in $\mathcal{F}_i$, ACAP have a single (deterministic) control function $D_1(t)$ which controls when to explore or exploit, while for each action (other learner) in $\mathcal{M}_{-i}$, ACAP have two (deterministic) control functions $D_2(t)$ and $D_3(t)$, where $D_2(t)$ controls when to train, $D_3(t)$ controls when to explore or exploit when there are enough trainings.

Marginal sample mean arm rewards, i.e., $\bar{r}_{k,C}^{i,d}(t)$, $k \in \mathcal{K}_i$, $C \in A_i^d(t)$, are calculated only based on the rewards collected at time steps when k is selected due to a type-d context in C being under-explored (i.e., at times when $N_{k,C^d(t')}^{i,d} \leq D_1(t')$ for $k \in \mathcal{F}_i$ or $N_{k,C^d(t')}^{i,d} \leq D_3(t')$ for $k \in \mathcal{M}_{-i}$, and $C^d(t') = C$, $t' < t$ in the adaptive partition). This way as the number of observations goes to infinity, it is guaranteed that

Train(t, k, d, $N_{1,k}^{i,d}$):

Select arm k.
Send current data $s_i(t)$ and type-d context $x_i^d(t)$ to learner k.
Receive prediction $\hat{y}_i(t) = f_{k,t}(s_i(t))$ from learner k, where $f_{k,t}$ is the classifier that k chooses to make the prediction which is selected using $x_i^d(t)$.
Receive true label $y_i(t)$ (send this also to learner k).
Compute reward $r_k(t) = \mathrm{I}(\hat{y}_i(t) = y_i(t)) - c_k^i$.
$N_{1,k,C^d(t)}^{i,d} + +$.

Explore(t, k, d, $N_{k,C^d(t)}^{i,d}$, $\bar{r}_{k,C^d(t)}^{i,d}$):

Select arm k.
Receive prediction $\hat{y}_i(t) = f_{k,t}(s_i(t))$.
Receive true label $y_i(t)$ (if $k \in \mathcal{M}_{-i}$, send this also to learner k).
Compute reward $r_k(t) = \mathrm{I}(\hat{y}_i(t) = y_i(t)) - c_k^i$.
$\bar{r}_{k,C^d(t)}^{i,d} = \frac{N_{k,C^d(t)}^{i,d} \bar{r}_{k,C^d(t)}^{i,d} + r_k(t)}{N_{k,C^d(t)}^{i,d} + 1}$.
$N_{k,C^d(t)}^{i,d} + +$.

Exploit(t, $\boldsymbol{N}^i$, $\hat{\boldsymbol{r}}^i$, $\mathcal{K}_i$):

Select arm $k \in \arg\max_{j\in\mathcal{K}_i} \left(\max_{d\in\mathcal{D}} \bar{r}_{j,C^d(t)}^{i,d}\right)$.
Receive prediction $\hat{y}_i(t) = f_{k,t}(s_i(t))$.
Receive true label $y_i(t)$ (if $k \in \mathcal{M}_{-i}$, send this also to learner k).
Compute reward $r_k(t) = \mathrm{I}(\hat{y}_i(t) = y_i(t)) - c_k^i$.

Fig. 6. Pseudocode of the training, exploration and exploitation modules.

$\bar{r}_{k,C}^{i,d}(t)$ converges to a number very close to the true marginal expected reward action k for contexts in C. This, together with the adaptive partitioning guarantees that the regret remains sublinear in time.

In addition to the exploration phase, which allows learners to build accurate estimates of the marginal expected rewards of their actions for their own contexts, the training phase required for actions $k \in \mathcal{M}_{-i}$ serves the purpose of helping the learners build accurate estimates of the marginal expected rewards for each other's contexts. If learner i selects another learner k, it cannot choose the classifier that is selected by that learner to produce the prediction. If the estimated marginal accuracies of classifiers of learner k are inaccurate, i's estimate of k's accuracy will be very different from the accuracy of k's marginally optimal classifier for i's context vector. Therefore, learner i uses the rewards from learner $k \in \mathcal{M}_{-i}$ to estimate the expected reward of learner k only if it believes that learner k estimated the accuracies of its own classifiers accurately.

In order for learner k to estimate the accuracies of its own classifiers accurately, if the number of context arrivals to learner k in set $C_i^d(t)$ is small, learner i *trains* learner k by sending its type-d context to k, receiving back the prediction of the classifier chosen by k and sending the true label at the end of that time slot to k so that k can compute the estimated accuracy of the classifier (in $\mathcal{F}_k$) it had chosen for i. In order to do this, learner i keeps two counters $N_{1,k,C}^{i,d}(t)$ and $N_{k,C}^{i,d}(t)$ for each $C \in \mathcal{A}_d^i(t)$, which are initially set to 0.

If $N_{1,k,C}^{i,d}(t) \leq D_2(t)$, then learner i trains learner k by sending its data $s_i(t)$, and context $x_i^d(t)$, receiving a prediction from learner k, and then sending the true label $y_i(t)$ to learner k so that learner k can update the estimated marginal accuracy

of the classifier in $\mathcal{F}_k$ it had chosen to make a prediction for learner i for type-d contexts. If $N^{i,d}_{1,k,C}(t) > D_2(t)$, for all $d \in \mathcal{D}$, this means that learner k is trained enough for all types of contexts so it will almost always select the classifier with the highest *marginal accuracy*, i.e., $\max_{d\in\mathcal{D}} \pi^d_f(x^d_i(t))$, $f \in \mathcal{F}_k$ when called by i.

To have sufficient observations from k before exploitation, i explores k when $N^{i,d}_{1,k,C}(t) > D_2(t)$ and $N^{i,d}_{k,C}(t) \leq D_3(t)$, and updates $N^{i,d}_{k,C}(t)$ and the sample mean marginal accuracy of learner k, which is the ratio of the total number of correct predictions to the total number of predictions k has made for i for contexts in hypercube C. Let

$$\mathcal{S}^{i,d}_{C^d_i(t)}(t) := \Big\{ f \in \mathcal{F}_i : N^{i,d}_{f,C^d_i(t)}(t) \leq D_1(t) \textbf{ or } j \in \mathcal{M}_{-i} : N^{i,d}_{1,j,C^d_i(t)}(t) \leq D_2(t) \textbf{ or } N^{i,d}_{j,C^d_i(t)}(t) \leq D_3(t) \Big\}, \quad (1)$$

and

$$\mathcal{S}^i_{\boldsymbol{C}_i(t)}(t) := \bigcup_{d\in\mathcal{D}} \mathcal{S}^{i,d}_{C^d_i(t)}(t).$$

If $S^i_{\boldsymbol{C}_i(t)}(t) \neq \emptyset$ then ACAP randomly selects an arm in $S^i_{\boldsymbol{C}_i(t)}(t)$ to train or explore, while if $S^i_{\boldsymbol{C}_i(t)}(t) = \emptyset$, ACAP selects an arm in

$$\arg\max_{k\in\mathcal{K}_i} \left(\max_{d\in\mathcal{D}} \bar{r}^{i,d}_{k,C^d_i(t)}(t) \right)$$

to exploit.

B. Analysis of the regret of ACAP

In this subsection we analyze the regret of ACAP and derive a sublinear upper bound on the regret, whose time order does not depend on D. We divide the regret $R_i(T)$ into three different terms. $R^e_i(T)$ is the regret due to trainings and exploitations by time T, $R^s_i(T)$ is the regret due to selecting suboptimal actions at exploitation steps by time T, and $R^n_i(T)$ is the regret due to selecting near-optimal actions in exploitation steps by time T. Using the fact that trainings, explorations and exploitations are separated over time, and linearity of expectation operator, we get $R_i(t) = R^e_i(T)+R^s_i(T)+R^n_i(T)$. In the following analysis, we will bound each part of the regret separately. Due to the limited space, proofs of the lemmas are given in our online appendix [24].

Lemma 1: **Regret of trainings and explorations in a hypercube.** Let $D_1(t) = D_3(t) = t^z \log t$ and $D_2(t) = F_{\max} t^z \log t$. Then, for any level l hypercube for type-d context the regret due to trainings and explorations by time t is bounded by

$$O(MF_{\max} t^z \log t).$$

For learner i let $\mu^d_k(x) := \pi^d_k(x) - c^i_k$, be the expected reward of arm $k \in \mathcal{K}_i$ for type-d context $x^d \in \mathcal{X}_d$. For each set of hypercubes $\boldsymbol{C} = (C^1, \ldots, C^D)$, let $k^*(\boldsymbol{C}) \in \mathcal{K}_i$ be the arm which is optimal for the center context of the type-d hypercube which has the highest expected reward among all types of contexts for $\boldsymbol{C}$, and let $d^*(\boldsymbol{C})$ be the type of the context for which arm $k^*(\boldsymbol{C})$ has the highest expected reward. Let $\overline{\mu}^d_{k,C^d} := \sup_{x\in C^d} \mu^d_k(x)$, $\underline{\mu}^d_{k,C^d} := \inf_{x\in C^d} \mu^d_k(x)$ $\overline{\mu}_{k,\boldsymbol{C}} := \max_{d\in\mathcal{D}} \overline{\mu}^d_{k,C^d}$, and $\underline{\mu}_{k,\boldsymbol{C}} := \max_{d\in\mathcal{D}} \underline{\mu}^d_{k,C^d}$, for $k \in \mathcal{K}_i$. When the set of active hypercubes of learner i is $\boldsymbol{C}$, the set of suboptimal arms is given by

$$\mathcal{L}^i_{\boldsymbol{C},B} := \left\{ k \in \mathcal{K}_i : \underline{\mu}_{k^*(\boldsymbol{C}),\boldsymbol{C}} - \overline{\mu}_{k,\boldsymbol{C}} > BL2^{-l_{\max}(\boldsymbol{C})\alpha} \right\},$$

where $B > 0$ is a constant and $l_{\max}(\boldsymbol{C})$ is the level of the highest level hypercube in $\boldsymbol{C}$. When the context vector is in $\boldsymbol{C}$, any arm that is not in $\mathcal{L}^i_{\boldsymbol{C},B}$ is a near-optimal arm.

Lemma 2: **Regret due to suboptimal arm selections.** Let $\mathcal{L}^i_{\boldsymbol{C},B}$, $B = 12/(L2^{-\alpha})+2$ denote the set of suboptimal arms for set of hypercubes $\boldsymbol{C}$. When ACAP is run with parameters $p > 0$, $2\alpha/p \leq z < 1$, $D_1(t) = D_3(t) = t^z \log t$ and $D_2(t) = F_{\max} t^z \log t$, the regret of learner i due to choosing suboptimal arms in $\mathcal{L}^i_{\boldsymbol{C}_i(t),B}$ at time slots $1 \leq t \leq T$ in exploitation steps, i.e., $R^s_i(T)$, is bounded by

$$O(MF_{\max} T^{z/2}).$$

Lemma 3: **Regret due to near-optimal learners choosing suboptimal classifiers.** Let $\mathcal{L}^i_{\boldsymbol{C},B}$, $B = 12/(L2^{-\alpha})+2$ denote the set of suboptimal actions for set of hypercubes $\boldsymbol{C}$. When ACAP is run with parameters $p > 0$, $2\alpha/p \leq z < 1$, $D_1(t) = D_3(t) = t^z \log t$ and $D_2(t) = F_{\max} t^z \log t$, for any set of hypercubes $\boldsymbol{C}$ that has been active and contained $\boldsymbol{x}_i(t')$ for some exploitation time slots $t' \in \{1, \ldots, T\}$, the regret due to a near optimal learner choosing a suboptimal classifier when called by learner i is bounded by

$$4(M-1)F_{\max}\beta_2.$$

The next lemma bounds the regret due to learner i choosing near optimal arms by time T.

Lemma 4: **Regret due to near-optimal arms.** Let $\mathcal{L}^i_{\boldsymbol{C},B}$, $B = 12/(L2^{-\alpha}) + 2$ denote the set of suboptimal actions for set of hypercubes $\boldsymbol{C}$. When ACAP is run with parameters $p > 0$, $2\alpha/p \leq z < 1$, $D_1(t) = D_3(t) = t^z \log t$ and $D_2(t) = F_{\max} t^z \log t$, the regret due to near optimal arm selections in $\mathcal{L}^i_{\boldsymbol{C}_i(t),B}$ at time slots $1 \leq t \leq T$ in exploitation steps is bounded above by

$$O\left(T^{\frac{1+p-\alpha}{1+p}}\right).$$

From Lemma 4, we see that the regret due to choosing near optimal arms increases with the parameter p that determines how much each hypercube will remain active, and decreases with α, which determines how similar is the expected accuracy of a classifier for similar contexts. Next, we combine the results from Lemmas 1, 2, 3 and 4 to obtain the regret bound for ACAP.

Theorem 1: Let $\mathcal{L}^i_{\boldsymbol{C},B}$, $B = 12/(L2^{-\alpha})+2$ denote the set of suboptimal actions for set of hypercubes $\boldsymbol{C}$. When ACAP is run with parameters $p = \frac{3\alpha+\sqrt{9\alpha^2+8\alpha}}{2}$, $z = 2\alpha/p < 1$, $D_1(t) = D_3(t) = t^z \log t$ and $D_2(t) = F_{\max} t^z \log t$, the regret of learner i by time T is upper bounded by

$$O\left(F_{\max} M T^{f_1(\alpha)} \log T\right),$$

where $f_1(\alpha) = (2+\alpha+\sqrt{9\alpha^2+8\alpha})/(2+3\alpha+\sqrt{9\alpha^2+8\alpha})$.

From the result of Theorem 1, it is observed that the regret increases linearly with the number of learners in the system and their number of classifiers (which $F_{\max}$ is an upper bound on). We note that the regret is the gap between the total expected reward of the optimal distributed policy that can be computed by a genie which knows the accuracy functions of every classifier, and the total expected reward of ACAP. Since the performance of optimal distributed policy never gets worse as more learners are added to the system or as more classifiers are introduced, the benchmark we compare our algorithm against with may improve. Therefore, the total reward of ACAP may improve even if the regret increases with M, $|\mathcal{F}_i|$ and $F_{\max}$.

V. Numerical Results

In this section, we numerically compare the performance ACAP with state–of–the–art online learning techniques in real-world datasets.

A. Data Sets

We consider four data sets, described below.

R1: Network Intrusion [25]–[28]. The network intrusion data set from UCI archive [25] consists of a series of TCP connection records, labeled either as normal connections or as attacks.

R2: Electricity Pricing [28]–[30]. The electricity pricing data set holds information for the Australian New South Wales electricity market. The binary label (up or down) identifies the change of the price relative to a moving average of the last 24 hours.

R3: Forest Cover Type [27], [30], [31]. The forest cover type data set from UCI archive [25] contains cartographic variables of four wilderness areas of the Roosevelt National Forest in northern Colorado. Each instance is classified with one of seven possible classes of forest cover type. Our task is to predict if an instance belong to the first class or to the other classes.

R4: Credit Card Risk Assessment [28], [32]. In the credit card risk assessment data set, used for the PAKDD 2009 Data Mining Competition [32], each instance contains information about a client that accesses to credit for purchasing on a specific retail chain. The client is labeled as good if he was able to return the credit in time, as bad otherwise.

B. Comparison with ensemble schemes

In this subsection we compare the performance of ACAP with state–of–the–art online ensemble learning techniques, listed in Table II. Different from ACAP which makes a prediction based on a single classifier at each time step, these techniques combine the predictions of all classifiers to make the final prediction. For a detailed description of the considered online ensemble learning techniques, we refer the reader to the cited references.

For each data set we consider a set of 8 logistic regression classifiers [39]. Each local classifier is pre–trained using an individual training data set and kept fixed for the whole simulation (except for OnAda, Wang, and DDD, in which the classifiers are retrained online). For ACAP we consider 4 learners, each of them possessing 2 of the 8 classifiers. For a fair comparison among ACAP and the considered ensemble schemes that do not deal with classification costs, we set c_k^i to 0 for all $k \in \mathcal{K}_i$. In all the simulations we consider a 3–dimensional context space. For the data sets **R1**–**R4** the first two context dimensions are the first two features whereas the last context dimension is the preceding label.

Table II lists the considered algorithms, the corresponding references, and their percentages of mis–classifications in the considered data sets. Importantly, in all the data sets ACAP is among the best schemes. This is not valid for the ensemble learning techniques. For example, WM is slightly more accurate than ACAP in **R1** and **R3**, it is slightly less accurate than ACAP in **R2**, but performs poorly in **R4**.

VI. Conclusion

In this paper we considered the problem of adaptively learning how to exploit a high dimensional context vector in Big Data stream mining. We proposed a novel learning framework which achieves sublinear regret with time order independent of the dimension of the context vector. Hence it can converge very fast to the marginally optimal benchmark, by learning the marginal accuracies of classifiers for each type of context. Our numerical results show that the proposed framework significantly improves the performance of Big Data systems.

References

[1] J. Gao, W. Fan, and J. Han, "On appropriate assumptions to mine data streams: Analysis and practice," in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 143–152.

[2] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 747–757, 2000.

[3] V. S. Tseng, C.-H. Lee, and J. Chia-Yu Chen, "An integrated data mining system for patient monitoring with applications on asthma care," in *Computer-Based Medical Systems, 2008. CBMS'08. 21st IEEE International Symposium on*. IEEE, 2008, pp. 290–292.

[4] R. Ducasse, D. S. Turaga, and M. van der Schaar, "Adaptive topologic optimization for large-scale stream mining," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 3, pp. 620–636, 2010.

[5] "Ibm smarter planet project," *http://www-03.ibm.com/software/products/en/infosphere-streams*.

[6] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, pp. 4–22, 1985.

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, p. 235256, 2002.

[8] A. Slivkins, "Contextual bandits with similarity information," in *24th Annual Conference on Learning Theory (COLT)*, 2011.

[9] J. Langford and T. Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1096–1103, 2007.

[10] J. B. Predd, S. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 23, no. 4, pp. 56–69, 2006.

[11] F. Pérez-Cruz and S. R. Kulkarni, "Robust and low complexity distributed kernel least squares learning in sensor networks," *Signal Processing Letters, IEEE*, vol. 17, no. 4, pp. 355–358, 2010.

[12] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

Abbreviation	Name of the Scheme	Reference	Performance			
			R1	R2	R3	R4
AM	Average Majority	[26]	3.07	41.8	29.5	34.1
Ada	Adaboost	[33]	3.07	41.8	29.5	34.1
OnAda	Fan's Online Adaboost	[34]	2.25	41.9	39.3	19.8
Wang	Wang's Online Adaboost	[35]	1.73	40.5	32.7	19.8
DDD	Diversity for Dealing with Drifts	[28]	1.15	44.0	23.9	19.9
WM	Weighted Majority algorithm	[36]	0.29	22.9	14.1	67.4
Blum	Blum's variant of WM	[37]	1.64	40.3	22.6	68.1
TrackExp	Herbster's variant of WM	[38]	0.52	23.0	14.8	22.0
ACAP	**Adaptive Contexts with Adaptive Partition**	our work	**0.71**	**5.8**	**19.2**	**19.9**

TABLE II
COMPARISON AMONG ACAP AND ENSEMBLE SCHEMES: PERCENTAGES OF MIS–CLASSIFICATIONS IN THE DATA SETS **R1**–**R4**.

[13] H. Zheng, S. R. Kulkarni, and H. Poor, "Attribute-distributed learning: models, limits, and algorithms," *Signal Processing, IEEE Transactions on*, vol. 59, no. 1, pp. 386–398, 2011.

[14] D. T. Y. Zhang, D. Sow and M. van der Schaar, "A fast online learning algorithm for distributed mining of bigdata," in *the Big Data Analytics workshop at SIGMETRICS 2013*, 2013.

[15] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *Signal Processing, IEEE Transactions on*, vol. 58, no. 10, pp. 5262–5276, 2010.

[16] C. Tekin and M. van der Schaar, "Distributed online big data classification using context information," in *Proc. of the 51st Annual Allerton Conference*, 2013.

[17] T. Lu, D. Pál, and M. Pál, "Contextual multi-armed bandits," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 485–492.

[18] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandits with linear payoff functions," in *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[19] P. Bühlmann and B. Yu, "Boosting with the l 2 loss: regression and classification," *Journal of the American Statistical Association*, vol. 98, no. 462, pp. 324–339, 2003.

[20] A. Lazarevic and Z. Obradovic, "The distributed boosting algorithm," in *Proc. of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2001, pp. 311–316.

[21] B. Chen, R. Jiang, T. Kasetkasem, and P. K. Varshney, "Channel aware decision fusion in wireless sensor networks," *Signal Processing, IEEE Transactions on*, vol. 52, no. 12, pp. 3454–3458, 2004.

[22] C. Perlich and G. Świrszcz, "On cross-validation and stacking: Building seemingly predictive models on random data," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 11–15, 2011.

[23] H. Kargupta, B. Park, D. Hershberger, and E. Johnson, "Collective data mining: A new perspective toward distributed data mining," *Advances in Distributed and Parallel Knowledge Discovery*, no. part II, pp. 131–174, 1999.

[24] C. Tekin, L. Canzian, and M. van der Schaar, "Context-adaptive big data stream mining - online appendix," *http://medianetlab.ee.ucla.edu/papers/Allerton14Appendix.pdf*, 2013.

[25] K. Bache and M. Lichman, "UCI machine learning repository," http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] J. A. Blackard, "Comparison of neural networks and discriminant analysis in predicting forest cover types," Ph.D. dissertation, Colorado State University, 1998.

[32] "PAKDD data mining competition 2009, credit risk assessment on a private label credit card application," http://sede.neurotech.com.br/PAKDD2009.

[33] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, Aug. 1997.

[34] 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.

[35] 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.

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

[37] 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.

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

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