Context-Adaptive Big Data Stream Mining - Online Appendix

Cem Tekin*, *Member, IEEE*, Luca Canzian, *Member, IEEE*, Mihaela van der Schaar, *Fellow, IEEE*
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. Learning is performed for all the possible types of contexts simultaneously, in parallel, rather than serially learning about different contexts at different times. This learning framework works for both single and multi-learner distributed data mining systems where each learner has access to a different set of classifiers. We theoretically bound the performance gap between our context-adaptive learning algorithm and a benchmark policy that knows everything about the accuracies of the classifiers and the arrival distribution of data, labels, and contexts. We show that curse of dimensionality can be avoided when classification decisions at each time instant are made depending on the best context from the set of contexts available at that time instant. Our numerical results illustrate that our algorithm outperforms most prior online learning algorithms, for which such online performance bounds have not been proven.**

Keywords: Stream mining, context-adaptive learning, distributed multi-user learning, contextual bandits, regret bounds, concept drift.

I. Introduction

A plethora of Big Data applications (network monitoring [1], surveillance [2], health monitoring [3], stock market prediction, intelligent driver assistance [4], etc.) 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 [5], [6]. For instance, it is well-known that such online stream mining problems need to cope with concept drift [27]. 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 dynamically changing data and the labels (feedback) received in the past, which type of contexts are the most relevant given a vector of available contexts at each time instance, and to choose the best classifier according to its estimated performance for that particular context, jointly at the same time. 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 contexts 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) [7]–[10] 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 best contexts. Using this approach, we are able to analytically bound the loss due to learning at any time instant. This is much stronger than heuristic approaches or other reinforcement learning based approaches in the literature for which no such bounds exist.

A key differentiating feature of the approach proposed in this paper is the focus on how the context information of the data stream can be effectively utilized to maximize the classification performance of an online data mining system. 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 contexts associated with the data is large (i.e., dimension of the context space 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 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 questions we aim to address in this paper by using our approach are:

- Does using the context information help improve the performance of the online stream mining system?
- The context information that comes along with the data can be high dimensional (a vector of different types of contexts). Should the decision of the learner depend on the entire context vector or a particular type of context? If the decision of the learner depends on a particular type of context should it be chosen initially or should it be adaptively learned over time, depending on the entire context vector? How should the adaptivity be performed?
- In a distributed learning environment, where multiple learners process different data, based on different contexts, how can these contexts be used? Can a learner exploit what other learners have learned in previous time slots?
- How should the learning process be performed when there is concept drift? How does concept drift affect the performance?

To optimize the run-time performance of the data mining system, we design online learning algorithms whose long-term average rewards converge to the best solution which can be obtained for the classification problem given complete knowledge of online data characteristics as well as their classifier accuracies and costs when applied to this data. The benchmark we compare against is a genie aided scheme, in which the genie knows classification accuracies of each classifier of each learner, and chooses the classifier which yields the highest expected accuracy for the best context in the set of available contexts at each time slot. We call the difference between the expected total reward (correct predictions minus cost) of a learner under the genie aided scheme and the expected total reward of the online learning algorithm used by the learner as the regret of the learner, and analytically derive a bound on the regret of our online learning algorithm under mild assumptions on the structure of the problem.

While we focus on adaptively learning the most relevant single context, with a simple modification, our algorithm can also be used to adaptively learn the best $N > 0$ contexts. The novel context-adaptive learning which we propose in this paper can be used for both single-learner stream mining systems as well as distributed multi-learner systems. Since the multi-learner system represents a super-set of the single-learner system, we focus on the multi-learner system.

The remainder of the paper is organized as follows. In Section II, we describe the related work. In Section III, we describe the decentralized data classification problem and the performance measure. Then, we consider the model with unknown system statistics without concept drift and propose a distributed online learning algorithm which learns to exploit the most relevant context adaptively over time in Section IV. In Section V, we extend our results such that our algorithm can track the most relevant context even under concept drift. Then, in Section VI, we give a detailed comparison of our algorithms with our previous work which do not adaptively learn the most relevant context. We provide numerical results and comparisons on the performance of our online learning algorithms in Section VII. Finally, the concluding remarks are given in Section VIII.

II. Related Work

Most of the prior work in stream mining is focused on learning from the unique data stream characteristics [11]–[19]. 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 assume no prior knowledge of the data and context arrival processes or the classifiers' accuracies. The learning is done in a non-Bayesian way, i.e., learners have no prior beliefs about the accuracies of classifiers and they do not have any distribution about the parameters of the system which they can use to perform Bayesian updating. Learning in a non-Bayesian way is appropriate in the considered stream mining systems since learners often do not have correct beliefs about the data dynamics.

Most of the prior work in (distributed) online stream mining provides algorithms which are asymptotically converging to an optimal or locally-optimal solution without providing any rates of convergence. On the contrary, we do not only prove convergence results, but we are also able to explicitly characterize the performance loss incurred at each time slot with respect to the optimal solution. In other words, we prove regret bounds that hold uniformly over time.

Some of the existing solutions (including [13], [14], [20]–[25]) propose ensemble learning techniques. In our work we only consider choosing the best classifier (initially unknown) from a set of classifiers that are accessible by learners. However, our proposed distributed learning methods can easily be adapted to perform ensemble learning. Interested readers can refer to [26], in which ensemble learning is used together with online learning with non-adaptive contexts. We provide a detailed comparison to our work in Table I.

Our contextual framework can also deal with concept drift [27]–[29]. In [27] a concept is formally defined as the probability distribution which generates the instances and the labels, and a concept drift refers to a change in such distribution. In this paper, we propose a new definition of concept drift: we define the concept drift as a variation of the accuracy of the classifiers. Notice that our definition is not completely different from the definition of [27]; in fact, if the underlying generating distribution varies, it is highly probable that the accuracies of the classifiers vary as well. However, large (small) variations of the generating distribution do not necessarily result in large (small) variations of the classifiers accuracies. Hence, if the classifiers represent an input of the stream mining problem (as in our case), our definition is more suitable. Finally, notice that our definition includes also situations in which the classifiers,

instead of the generating distribution, change in time (e.g., online classifiers), whereas [27] does not consider this situation as a concept drift.

Other than distributed data mining, our learning framework can be applied to any problem that can be formulated as a decentralized contextual bandit problem [30]. Contextual bandits have been studied before in [9], [10] 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 best action according to the most relevant type of context among all D types of contexts. Therefore, each agent should adaptively learn the most relevant context.

Our prior work has shown that the performance of existing learning algorithms usually depends on the dimension of the context space [30], [31]. When the dimension of the context space is large, the convergence rate of these algorithms to the optimal average reward becomes very slow. However, in real-time stream mining applications, short time performance is as important as the long term performance. In addition to this, although there may be many types of meta-data to exploit, the types of meta-data that is highly correlated with the data and label is usually small. In other words, although the number and types of contexts that the learning algorithm can use may be large, the relevant contexts represent at each time only a small subset of the available/possible contexts. However, the convergence speed of the algorithms we propose in this paper is independent of this dimension. By learning the most relevant context over time, in this paper, we are able to design a learning algorithm for any $D > 0$ dimensional context space, with a convergence rate as fast as the the convergence rate of the algorithms in prior work in a one dimensional context space.

III. Problem Formulation

Even though the focus of this paper is to highlight the importance of exploiting context and adaptively learning the best context to exploit, we will present our results in the most general form, involving multiple distributed learners operating on different data and context streams. Then, both the algorithm and the results for the centralized problem with a single learner which learns about multiple classifiers follows by letting the set of other learners be equal to the emptyset. Thus, we will first give the formulation, learning algorithms and results for multiple learners, and then we will explain how these algorithms work when there is only one learner.

The system model is shown in Fig. 1 and 2. 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 set[1] of learner i is $\mathcal{K}_i := \mathcal{F}_i \cup \mathcal{M}_{-i}$.

[1]In sequential online learning literature [7], [8], an action is also called an arm (or an alternative).

Throughout the paper we use index f to denote an element of $\mathcal{F}$, j_i to denote learners in $\mathcal{M}_{-i}$, f_i to denote an element of $\mathcal{F}_i$, 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. In the next two subsections we will define how the data, label and context vector is generated, and then based on this, define the classifier accuracies.

A. Stationary data, label and context with similarity property

In this subsection, we assume that 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 $\pi_f(\boldsymbol{x}) := E[I(f(s_i(t)) = y_i(t))] = \int_{(s,y)\in\mathcal{S}\times\mathcal{Y}} I(f(s_i(t)) = y_i(t))dG_{\boldsymbol{x}}(s,y)$ be the expected accuracy of classifier $f \in \mathcal{F}$, where $f(s_i(t))$ is the prediction of classifier f on the data, $I(\cdot)$ is the indicator function which is equal to 1 if the statement inside is true and 0 otherwise, and the expectation $E[\cdot]$ is taken with respect to distribution $G_{\boldsymbol{x}}$. Let $\boldsymbol{x}^{-d} := (x^1, \ldots, x^{d-1}, x^{d+1}, \ldots, x^D)$ and $((\boldsymbol{x}')^{-d}, x^d) = (x'^1, \ldots, x'^{d-1}, x^d, x'^{d+1}, \ldots, x'^D)$. Then, the expected accuracy of f based only 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 instance involving joint distribution over data, label and contexts, and the classifiers $f \in \mathcal{F}$ has similarity property when each classifier has similar accuracies for similar contexts.

Definition 1: **Stationary distribution with similarity.** 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 stationary 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

[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.

	[13], [18], [23]–[25]	[17], [19]	[15]	[26], [30], [31]	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.

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. Data, label and context with gradual concept drift and similarity property

In this subsection, we assume that the joint distribution over data, label and contexts changes gradually over time, hence denote the joint distribution at time t by J_t and the corresponding conditional distribution over $\mathcal{S} \times \mathcal{Y}$ and marginal distribution over $\mathcal{X}$ by $G_{\boldsymbol{x},t}$ and H_t, respectively. Classification accuracies are defined similarly to the previous subsection. We have $\pi_{f,t}(\boldsymbol{x}) := E_t[I(f(s_i(t)) = y_i(t))] = \int_{(s,y)\in\mathcal{S}\times\mathcal{Y}} I(f(s_i(t)) = y_i(t)) dG_{\boldsymbol{x},t}(s,y)$ and $\pi^d_{f,t}(x^d) := \int_{(\boldsymbol{x}')^{-d}} \pi_{f,t}((\boldsymbol{x}')^{-d}, x^d) dH_t((\boldsymbol{x}'^{-d}), x^d)$, where $E_t[.]$ denotes the expectation with respect to distribution $G_{\boldsymbol{x},t}$.

We say that the problem instance involving joint distributions $J_1, J_2, \ldots, J_T$ over data, label and contexts, and the classifiers $f \in \mathcal{F}$ have similarity property with gradual concept drift when the distance between the accuracy of each classifier for similar contexts between two time slots t and t' is non-decreasing in the distance between t and t'.

Definition 2: **Gradual concept drift with similarity.** If the sequence of joint distributions $J_1, J_2, \ldots, J_T$ over $\mathcal{S} \times \mathcal{Y} \times \mathcal{X}$ are such that for each $t, t' \in \{1, 2, \ldots, T\}$, $f \in \mathcal{F}$ and $d \in \mathcal{D}$, there exists a minimum $\alpha > 0$, and a minimum $0 < L < \tau^\alpha$ such that for all $x^d, (x')^d \in \mathcal{X}_d$, we have $|\pi^d_{f,t}(x^d) - \pi^d_{f,t}((x')^d)| \leq L \left(|x^d - (x')^d|^2 + |\frac{t'}{\tau} - \frac{t}{\tau}|^2\right)^{\alpha/2}$, where $\tau > 0$ is a parameter that quantifies the stability of the concept, then we call $J_1, J_2, \ldots, J_T$, *gradually drifting distributions with similarity*. When τ is large the concept is more stable. We assume that the learners know τ, while our results will also hold when learners only know a lower bound on τ.

In Definition 2, the upper bound on the value of L is needed due to the fact that increments in time are discrete. Otherwise, for any sequence of joint distributions $J_1, J_2, \ldots, J_T$ which can be very different from each other, if each individual distribution satisfies the condition given in Definition 1, then there exists an L large enough such that the condition in Definition 2 is satisfied.

A simple example of sequence of distributions that is gradually drifting with similarity is the following. Let $\mathcal{S} = \mathcal{X} = \mathcal{Y} = \{0, 1\}$. Assume that context is the data itself and at each time data is independently drawn from a distribution with $P(s_i(t) = 1) = p_1$ and $P(s_i(t) = 0) = 1 - p_1$. True label is conditionally dependent on the data with distributions $P(y_i(t) = 0|s_i(t) = 0) = g_0(t)$ and $P(y_i(t) = 1|s_i(t) = 1) = g_1(t)$, where g_0 and g_1 are such that $|g_l(t) - g_l(t')| \leq L'(t - t')^{\alpha'}/\tau^{\alpha'}$, $l \in \{0, 1\}$, $L' < \tau^{\alpha'}$ and $\alpha' > 0$. Let f be a classifier such that when $x = 0$ it produces label $y = 0$, and when $x = 1$ it produces label $y = 1$. Then, $\pi_f(x)$ satisfies the condition given in Definition 2 with $L = L'$ and $\alpha = \alpha'$.

C. Unknowns, actions and rewards

In our problem, the unknowns for learner i when there is no concept drift 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) $\pi^d_f(x^d)$, $f \in \mathcal{F}_i$, $x^d \in \mathcal{X}_d$, $d \in 1, \ldots, D$. When there is concept drift, some of these unknowns are also a function of time. 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., $F_{\max} \geq |\mathcal{F}_{j_i}|$[4], for all $j_i \in \mathcal{M}_{-i}$.

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 $f_i \in \mathcal{F}_i$ incurs a cost $c^i_{f_i} \geq 0$. For example, if the application is delay critical this can be the delay cost, or this can represent the computational cost and power consumption associated with calling a classifier, or it can be zero if there is no cost. We assume that a learner can only call one classifier for each input data in order to label it. This is a reasonable assumption when costs of calling classifiers are high. Our results can be generalized to the case when multiple classifiers are called at each time. However, for simplicity of analysis and analytical tractability, we focus on the case of calling a single classifier. 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 $j_i \in \mathcal{M}_{-i}$ a cost of $c^i_{j_i}$ is incurred by learner i[5]. Since the costs are bounded, without loss of generality we assume that costs are normalized, i.e., $c^i_k \in [0, 1]$ for all $k \in \mathcal{K}_i$[6]. The learners are cooperative which implies that learner $j_i \in \mathcal{M}_{-i}$ 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 $j_i \in \mathcal{M}_{-i}$, learner i will return a label to j_i. We do not consider the effect of this on i's learning rate;

[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_i. Since the learners are cooperative, j_i 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_i will select a classifier from $\mathcal{F}_{j_i}$, 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.

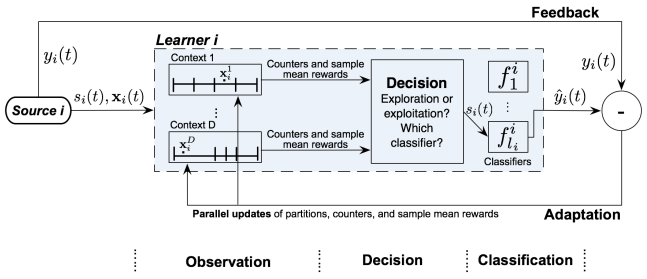

Fig. 1. Operation of learner i during a time slot when it chooses one of its own classifiers.

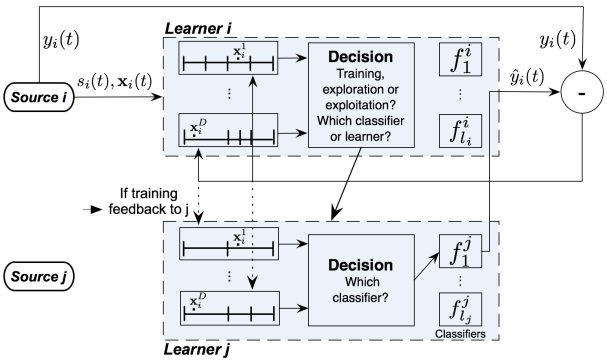

Fig. 2. Operation of learner i during a time slot when it chooses learner j.

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. If we assume that $c^i_{j_i}$ also captures the cost to learner j_i to classify and send the label back to learner i, then maximizing i's own expected accuracy minus cost corresponds to maximizing the sum of expected accuracy minus cost of all learners.

We assume that each classifier produces a binary label[7], thus $\mathcal{Y} = \{0, 1\}$. For a learner $j_i \in \mathcal{M}_{-i}$ its expected accuracy for a type-d context x^d is equal to the expected accuracy of its best classifier, i.e., $\pi^d_{j_i}(x^d) = \max_{k_{j_i} \in \mathcal{F}_{j_i}} \pi^d_{k_{j_i}}(x^d)$. 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 subsections.

D. Optimal Classification with Complete Information

Our benchmark when evaluating the performance of the learning algorithms is the optimal solution which selects the arm in $\mathcal{K}_i$ with the highest accuracy minus cost (i.e., reward) for the best type of context for learner i given the context vector $\boldsymbol{x}_i(t)$ at time t. We assume that the costs are normalized so the tradeoff between accuracy and cost is captured without using weights. Specifically, the optimal solution we compare against is given by $k^*_i(\boldsymbol{x}) = \arg\max_{k \in \mathcal{K}_i} \left(\max_{x^d \in \boldsymbol{x}} \pi^d_k(x^d) - c^i_k \right), \quad \forall \boldsymbol{x} \in \mathcal{X}$, when there is no concept drift, and by $k^*_i(\boldsymbol{x}, t) = \arg\max_{k \in \mathcal{K}_i} \left(\max_{x^d \in \boldsymbol{x}} \pi^d_{k,t}(x^d) - c^i_k \right), \quad \forall \boldsymbol{x} \in \mathcal{X}$, when there is concept drift. Knowing the optimal solution means that learner i knows the classifier in $\mathcal{F}$ that yields the highest expected accuracy for each $x^d \in \mathcal{X}_d$, $d \in \mathcal{D}$. Even when everything is known, choosing the best classifier for each context $\boldsymbol{x}$ requires to evaluate the accuracy minus cost for each context and is computationally intractable, because the context space $\mathcal{X}$ has infinitely many elements.

[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.

E. The Regret of Learning

Simply, the regret is the loss incurred due to the unknown system dynamics. Regret of a learning algorithm α which selects an action $\alpha_t(\boldsymbol{x}_i(t)) \in \mathcal{K}_i$ at time t for learner i is defined with respect to the best arm $k^*_i(\boldsymbol{x}_i(t))$ ($k^*_i(\boldsymbol{x}_i(t), t)$ when there is concept drift) at time t. The regret of a learning algorithm for learner i when there is no concept drift is given by $R_i(T) := \sum_{t=1}^{T} \left(\pi_{k^*_i(\boldsymbol{x}_i(t))}(\boldsymbol{x}_i(t)) - c^i_{k^*_i(\boldsymbol{x}_i(t))} \right) - E\left[\sum_{t=1}^{T} (I(\hat{y}^i_t(\alpha_t(\boldsymbol{x}_i(t))) = y_i(t)) - c^i_{\alpha_t(\boldsymbol{x}_i(t))}) \right]$, where $\hat{y}^i_t(.)$ denotes the prediction of the action 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 optimal 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 optimal solution in terms of the average reward. When there is concept drift, the regret can be defined in a similar way with respect to the optimal policy $k^*_i(\boldsymbol{x}_i(t), t)$. In general, the regret is linear in T when there is concept drift.

IV. Adaptive Contexts with Adaptive Partition For Stationary Data and Context

In this section we propose an online learning algorithm with sublinear regret when there is no concept drift and the condition in Definition 1 holds. Different from prior explore-exploit learning algorithms, our algorithm uses a three-phased learning structure which includes training, exploration and exploitation phases. The novel training phase helps a learner to teach others how to choose good classifiers, so that when asked to make predictions, they will learn to choose their best classifiers. 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 accuracy 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 best context from the context vector given to the learner at that time slot, using the sample mean estimates of the expected accuracies of the arms based on previous observations relevant to current contexts, which are computed separately for each type of context in the context vector. These sample mean accuracies

are updated in parallel for each context in the context vector, while the decision made at an exploitation step depends on the context which offers the highest estimated accuracy for the best arm for the context vector. We call the context which the decision is based on at time t 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. 3, and the initialization, training, exploration and exploitation modules are given in Fig. 4 and Fig. 5.

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 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 arm in $\mathcal{F}_i$, ACAP have a single (deterministic) control function $D_1(t)$ which controls when to explore or exploit, while for each arm 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 or not, $D_3(t)$ controls when to explore or exploit when there are enough trainings. When type-d context is selected as the *main context* at time t, for an arm $k \in \mathcal{F}_i$, all the observations up to time t in hypercube $C_i^d(t)$ are used by learner i to estimate the expected reward of that arm. This estimation is different for $k \in \mathcal{M}_{-i}$. This is because learner i cannot choose the arm that is selected by learner k when called by i. If the estimated rewards of arms of learner k are inaccurate, i's estimate of k's reward will be very different from the expected reward of k's optimal arm for i's context vector. Therefore, learner i uses the rewards from learner $j_i \in \mathcal{M}_{-i}$ to estimate the expected reward of learner j_i only if it believes that learner j_i estimated the expected rewards of its own arms accurately. In order for learner j_i to estimate the rewards of its own arms accurately, if the number of

[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, $\boldsymbol{N}_k^i = (N_{k,C^d(t)}^{i,d})_{d \in \mathcal{D}}$, $k \in \mathcal{K}_i$, $\boldsymbol{N}_{1,k}^i = (N_{1,k,C^d(t)}^{i,d})_{d \in \mathcal{D}}$, $k \in \mathcal{M}_{-i}$, $\boldsymbol{N}^i = (\boldsymbol{N}_k^i)_{k \in \mathcal{K}_i}$.
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, $\boldsymbol{N}_k^i$, $\bar{\boldsymbol{r}}_k^i$)
  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**
    Obtain $N_{C^d(t)}^{k,d}(t)$ from learner k.
    if $N_{C^d(t)}^{k,d}(t) = 0$ **then**
      Ask k to create hypercube $C^d(t)$ for its type-d context, set $N_{1,k,C^d(t)}^{i,d} = 0$
    else
      Set $N_{1,k,C^d(t)}^{i,d} = N_{C^d(t)}^{k,d}(t) - N_{k,C^d(t)}^{i,d}$
    end if
    if $N_{1,k,C^d(t)}^{i,d} \leq D_2(t)$ **then**
      Run **Train**(t, k, $\boldsymbol{N}_{1,k}^i$)
    else
      Go to line 7
    end if
  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, $\boldsymbol{N}_k^i$, $\bar{\boldsymbol{r}}_k^i$)
  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. 3. 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. 4. Pseudocode of the initialization module.

Train(t, k, $\boldsymbol{N}^i_{1,k}$):
1: Select arm k.
2: Send current data and context vector to learner k.
3: Receive prediction $\hat{y}_k(s_i(t), \boldsymbol{x}_i(t))$ from learner k.
4: Receive true label $y_i(t)$ (send this also to learner k).
5: Compute reward $r_k(t) = I(\hat{y}_k(s_i(t), \boldsymbol{x}_i(t)) = y_i(t)) - c^i_k$.
6: $N^{i,d}_{k,C^d(t)} + +$ for $d \in \mathcal{D}$.

Explore(t, k, $\boldsymbol{N}^i_k$, $\bar{\boldsymbol{r}}^i_k$):
1: Select arm k.
2: Receive prediction $\hat{y}_k(s_i(t), \boldsymbol{x}_i(t))$.
3: Receive true label $y_i(t)$ (if $k \in \mathcal{M}_{-i}$, send this also to learner k).
4: Compute reward $r_k(t) = I(\hat{y}_k(s_i(t), \boldsymbol{x}_i(t)) = y_i(t)) - c^i_k$.
5: $\bar{r}^{i,d}_{k,C^d(t)} = \frac{N^{i,d}_{k,C^d(t)} \bar{r}^{i,d}_{k,C^d(t)} + r_k(t)}{N^{i,d}_{k,C^d(t)} + 1}$, $d \in \mathcal{D}$.
6: $N^{i,d}_{k,C^d(t)} + +$, $d \in \mathcal{D}$.

Exploit(t, $\boldsymbol{N}^i$, $\hat{\boldsymbol{r}}^i$, $\mathcal{K}_i$):
1: Select arm $k \in \arg\max_{j \in \mathcal{K}_i} \left(\max_{d \in \mathcal{D}} \bar{r}^{i,d}_{j,C^d(t)} \right)$.
2: Receive prediction $\hat{y}_k(s_i(t), \boldsymbol{x}_i(t))$.
3: Receive true label $y_i(t)$ (if $k \in \mathcal{M}_{-i}$, send this also to learner k).
4: Compute reward $r_k(t) = I(\hat{y}_k(s_i(t), \boldsymbol{x}_i(t)) = y_i(t)) - c^i_k$.
5: $\bar{r}^{i,d}_{k,C^d(t)} = \frac{N^{i,d}_{k,C^d(t)} \bar{r}^{i,d}_{k,C^d(t)} + r_k(t)}{N^{i,d}_{k,C^d(t)} + 1}$, $d \in \mathcal{D}$.
6: $N^{i,d}_{k,C^d(t)} + +$, $d \in \mathcal{D}$.

Fig. 5. Pseudocode of the training, exploration and exploitation modules.

context arrivals to learner j_i in set $C^d_i(t)$ is small, learner i *trains* learner j_i by sending its context to j_i, receiving back the prediction of the classifier chosen by j_i and sending the true label at the end of that time slot to j_i so that j_i can compute the estimated accuracy of the classifier (in $\mathcal{F}_{j_i}$) it had chosen for i. In order to do this, learner i keeps two counters $N^{i,d}_{1,j_i,C}(t)$ and $N^{i,d}_{2,j_i,C}(t)$ for each $C \in \mathcal{A}^i_d(t)$, which are initially set to 0. At the beginning of each time slot for which $N^{i,d}_{1,j_i,C}(t) \leq D_2(t)$, learner i asks j_i to send it $N^{j_i,d}_C(t)$ which is the number of type-d context arrivals to learner j_i in C from the activation of C by learner j_i to time t, including the contexts sent by learner i and by other learners to learner j_i. If C has not been activated by j_i yet, then it sends $N^{j_i,d}_C(t) = 0$ and activates the hypercube C for its type-d context. Then learner i sets $N^{i,d}_{1,j_i,C}(t) = N^{j_i,d}_C(t) - N^{i,d}_{2,j_i,C}(t)$ and checks again if $N^{i,d}_{1,j_i,C}(t) \leq D_2(t)$ for some $d \in \mathcal{D}$. If so, then it trains learner j_i by sending its data and context stream $s_i(t), \boldsymbol{x}_i(t)$, receiving a prediction from learner j_i, and then sending the true label $y_i(t)$ to learner j_i so that learner j_i can update the estimated accuracy of the classifier in $\mathcal{F}_j$ it had chosen to make a prediction for learner i. If $N^{i,d}_{1,j_i,C}(t) > D_2(t)$, for all $d \in \mathcal{D}$, this means that learner j_i is trained enough for all types of contexts so it will almost always select its optimal arm when called by i. Therefore, i will only use observations when $N^{i,d}_{1,j_i,C}(t) > D_2(t)$ to estimate the expected reward of learner j_i for type-d contexts. To have sufficient observations from j_i before exploitation, i explores j_i when $N^{i,d}_{1,j_i,C}(t) > D_2(t)$ and $N^{i,d}_{2,j_i,C}(t) \leq D_3(t)$, and updates $N^{i,d}_{2,j_i,C}(t)$ and the sample mean accuracy of learner j_i, which is the ratio of the total number of correct predictions to the total number of predictions j_i has made for i for contexts in hypercube C. For simplicity of notation in the pseudocode of ACAP we let $N^{i,d}_{j_i,C}(t) := N^{i,d}_{2,j_i,C}(t)$ for $j_i \in \mathcal{M}_{-i}$. Let $\mathcal{S}^{i,d}_{C^d_i(t)}(t) := \{k_i \in \mathcal{F}_i$ **such that** $N^{i,d}_{k_i,C^d_i(t)}(t) \leq D_1(t)$ **or** $j_i \in \mathcal{M}_{-i}$ **such that** $N^{i,d}_{1,j_i,C^d_i(t)}(t) \leq D_2(t)$ **or** $N^{i,d}_{2,j_i,C^d_i(t)}(t) \leq D_3(t) \}$, and $\mathcal{S}^i_{\boldsymbol{C}_i(t)}(t) = \cup_{d \in \mathcal{D}} \mathcal{S}^{i,d}_{C^d_i(t)}(t)$. 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, where $\bar{r}^{i,d}_{k,C^d_i(t)}(t)$ is the sample mean of the rewards collected from arm k in time slots for which the type-d context is in $C^d_i(t)$ from the activation of $C^d_i(t)$ by learner i to time t for $k \in \mathcal{F}_i$, and it is the sample mean of the rewards collected from exploration and exploitation steps of arm k in time slots for which the type-d context is in $C^d_i(t)$ from the activation of $C^d_i(t)$ to time t for $k \in \mathcal{M}_{-i}$.

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. 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. Let $\beta_2 := \sum_{t=1}^{\infty} 1/t^2 = \pi^2/6$. For a set A, A^c denotes the complement of that set. . We start with a simple lemma which gives an upper bound on the highest level hypercube that is active at any time t.

Lemma 1: **A bound on the level of active hypercubes.** All the active hypercubes $\mathcal{A}^d_i(t)$ for type-d contexts at time t have at most a level of $(\log_2 t)/p + 1$.

Proof: Let $l+1$ be the level of the highest level active hypercube. We must have $A \sum_{j=0}^{l} 2^{pj} < t$, otherwise the highest level active hypercube will be less than $l+1$. We have for $t/A > 1$, $A \frac{2^{p(l+1)}-1}{2^p - 1} < t \Rightarrow 2^{pl} < \frac{t}{A} \Rightarrow l < \frac{\log_2 t}{p}$. ■

The following lemma bounds the regret due to trainings and explorations in a level l hypercube for a type-d context.

Lemma 2: **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 above by $2(|\mathcal{F}_i| + (M-1)(F_{\max}+1))(t^z \log t + 1)$.

Proof: This directly follows from the number of trainings and explorations that are required before any arm can be exploited (see definition of $S^i_{\boldsymbol{C}_i(t)}(t)$). If the prediction at any training or exploration step is incorrect or a high cost arm is chosen, learner i loses at most 2 from the highest realized reward it could get at that time slot, due to the fact an incorrect prediction will result in one unit of loss and the cost of an action can at most be one. ■

Lemma 2 states that the regret due to trainings and explorations increases exponentially with z, which controls the rate of learning. For learner i let $\mu^d_k(x) := \pi^d_k(x) - c^i_k$, i.e., 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. In the next lemma we bound the regret due to choosing a suboptimal arm in the exploitation steps.

Lemma 3: **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 above by $2(1+D)\beta_2|\mathcal{F}_i| + 8(M-1)F_{\max}\beta_2 T^{z/2}/z$.

Proof: Let Ω denote the space of all possible outcomes, and w be a sample path. The event that the ACAP exploits when $\boldsymbol{x}_i(t) \in \boldsymbol{C}$ is given by $\mathcal{W}^i_{\boldsymbol{C}}(t) := \{w : S^i_{\boldsymbol{C}}(t) = \emptyset, \boldsymbol{x}_i(t) \in \boldsymbol{C}, \boldsymbol{C} \in \mathcal{A}_i(t)\}$. We will bound the probability that ACAP chooses a suboptimal arm for learner i in an exploitation step when i's context vector is in the set of active hypercubes $\boldsymbol{C}$ for any $\boldsymbol{C}$, and then use this to bound the expected number of times a suboptimal arm is chosen by learner i in exploitation steps using ACAP. Recall that reward loss in every step in which a suboptimal arm is chosen can be at most 2.

Let $\mathcal{V}^i_{k,\boldsymbol{C}}(t)$ be the event that a suboptimal arm k is chosen for the set of hypercubes $\boldsymbol{C}$ by learner i at time t. For $k \in \mathcal{K}_i \cap \mathcal{F}_i$, let $\mathcal{E}^i_{k,\boldsymbol{C}}(t)$ be the set of rewards collected by learner i from arm k in time slots when the context vector of learner i is in the active set $\boldsymbol{C}$ by time t. For $j_i \in \mathcal{K}_i \cap \mathcal{M}_{-i}$, let $\mathcal{E}^i_{j_i,\boldsymbol{C}}(t)$ be the set of rewards collected from selections of learner j_i in time slots $t' \in \{1, \ldots, t\}$ for which $N^i_{1,j_i,l}(t') > D_2(t')$ and the context vector of learner i is in the active set $\boldsymbol{C}$ by time t. Let $\mathcal{B}^i_{j_i,\boldsymbol{C}}(t)$ be the event that at most t^ϕ samples in $\mathcal{E}^i_{j_i,\boldsymbol{C}}(t)$ are collected from suboptimal arms of learner j_i. For $k \in \mathcal{K}_i \cap \mathcal{F}_i$ let $\mathcal{B}^i_{k,\boldsymbol{C}}(t) := \Omega$. In order to facilitate our analysis of the regret, we generate two different artificial independent and identically distributed (i.i.d.) processes to bound the probabilities related to deviation of sample mean reward estimates $\bar{r}^{i,d}_{k,C^d}(t)$, $k \in \mathcal{K}_i$, $d \in \mathcal{D}$ from the expected rewards, which will be used to bound the probability of choosing a suboptimal arm. The first one is the *best* process in which rewards are generated according to a bounded i.i.d. process with expected reward $\overline{\mu}^d_{k,C^d}$, the other one is the *worst* process in which the rewards are generated according to a bounded i.i.d. process with expected reward $\underline{\mu}^d_{k,C^d}$. Let $\bar{r}^{b,i,d}_{k,C^d}(t)$ denote the sample mean of the t samples from the best process and $\bar{r}^{w,i,d}_{k,C^d}(t)$ denote the sample mean of the t samples from the worst process. We have for any $k \in \mathcal{L}^i_{\boldsymbol{C},B}$

$$
\begin{aligned}
&P\left(\mathcal{V}^i_{k,\boldsymbol{C}}(t), \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \\
&\leq P\left(\max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) \geq \overline{\mu}_{k,\boldsymbol{C}} + H_t, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \\
&+ P\left(\max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) \geq \bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t))\right. \\
&-2t^{\phi-1}, \max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) < \overline{\mu}_{k,\boldsymbol{C}} + L2^{-l_{\max}(\boldsymbol{C})\alpha} \\
&+H_t + 2t^{\phi-1}, \bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t)) \\
&\left.> \underline{\mu}_{k^*(\boldsymbol{C}),\boldsymbol{C}} - L2^{-l_{\max}(\boldsymbol{C})\alpha} - H_t, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \qquad (1) \\
&+ P\left(\bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t)) \leq \underline{\mu}_{k^*(\boldsymbol{C}),\boldsymbol{C}} - H_t\right. \\
&\left.+2t^{\phi-1}, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) + P((\mathcal{B}^i_{k,\boldsymbol{C}}(t))^c),
\end{aligned}
$$

where $H_t > 0$. In order to make the probability in (1) equal to 0, we need

$$4t^{\phi-1} + 2H_t \leq (B-2)L2^{-l_{\max}(\boldsymbol{C})\alpha}. \qquad (2)$$

By Lemma 1, (2) holds when

$$4t^{\phi-1} + 2H_t \leq (B-2)L2^{-\alpha}t^{-\alpha/p}. \qquad (3)$$

For $H_t = 4t^{\phi-1}$, $\phi = 1 - z/2$, $z \geq 2\alpha/p$ and $B = 12/(L2^{-\alpha}) + 2$, (3) holds by which (1) is equal to zero. Also by using a Chernoff-Hoeffding bound we can show that

$$P\left(\max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) \geq \overline{\mu}_{k,\boldsymbol{C}} + H_t, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \leq D/t^2,$$

and

$$
\begin{aligned}
&+ P\left(\bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t)) \leq \underline{\mu}_{k^*(\boldsymbol{C}),\boldsymbol{C}} - H_t\right. \\
&\left.+2t^{\phi-1}, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \leq 1/t^2.
\end{aligned}
$$

We also have $P(\mathcal{B}^i_{k,\boldsymbol{C}}(t)^c) = 0$ for $k \in \mathcal{F}_i$ and $P(\mathcal{B}^i_{j_i,\boldsymbol{C}}(t)^c) \leq E[X^i_{j_i,\boldsymbol{C}}(t)]/t^\phi \leq 2F_{\max}\beta_2 t^{z/2-1}$. for $j_i \in \mathcal{M}_{-i}$, where $X^i_{j_i,\boldsymbol{C}}(t)$ is the number of times a suboptimal arm of learner j_i is selected when learner i calls j_i in exploration and exploitation phases in time slots when the context vector of i is in the set of hypercubes $\boldsymbol{C}$ which are active by time t. Combining all of these we get $P\left(\mathcal{V}^i_{k_i,\boldsymbol{C}}(t), \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \leq (1+D)/t^2$, for $k \in \mathcal{F}_i$ and $P\left(\mathcal{V}^i_{j_i,\boldsymbol{C}}(t), \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \leq (1+D)/t^2 + 2F_{\max}\beta_2 t^{z/2-1}$, for $j_i \in \mathcal{M}_{-i}$. We get the final bound by summing these probabilities from $t = 1$ to T. ■

In the next lemma we bound the regret due to near optimal learners choosing their suboptimal classifiers when called by learner i in exploitation steps when the context vector of learner i belongs to is $\boldsymbol{C}$.

Lemma 4: **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 upper bounded by $4(M-1)F_{\max}\beta_2$.

Proof: Let $X^i_{j_i,\boldsymbol{C}}(T)$ denote the random variable which is the number of times a suboptimal arm for learner $j_i \in \mathcal{M}_{-i}$ is chosen in exploitation steps of i when $\boldsymbol{x}_i(t')$ is in set $\boldsymbol{C} \in \mathcal{A}_i(t')$ for $t' \in \{1, \ldots, T\}$. It can be shown that $E[X^i_{j_i,\boldsymbol{C}}(T)] \leq 2F_{\max}\beta_2$. Thus, the contribution to the regret from suboptimal arms of j_i is bounded by $4F_{\max}\beta_2$. We get the final result by considering the regret from all $M-1$ other learners. ∎

The following lemma bounds the one-step regret to learner i from choosing near optimal arms. This lemma is used later to bound the total regret from near optimal arms.

Lemma 5: **One-step regret due to near-optimal arms for a set of hypercubes.** 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 $\boldsymbol{C}$, the one-step regret of learner i from choosing one of its near optimal classifiers is bounded above by $BL2^{-l_{\max}(\boldsymbol{C})\alpha}$, while the one-step regret of learner i from choosing a near optimal learner which chooses one of its near optimal classifiers is bounded above by $2BL2^{-l_{\max}(\boldsymbol{C})\alpha}$.

Proof: At time t if $\boldsymbol{x}_i(t) \in \boldsymbol{C} \in \mathcal{A}_i(t)$, the one-step regret of any near optimal arm of any near optimal learner $j_i \in \mathcal{M}_{-i}$ is bounded by $2BL2^{-l_{\max}(\boldsymbol{C})\alpha}$ by the definition of $\mathcal{L}^i_{\boldsymbol{C},B}$. Similarly, the one-step regret of any near optimal arm $k \in \mathcal{F}_i$ is bounded by $BL2^{-l_{\max}(\boldsymbol{C})\alpha}$. ∎

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

Lemma 6: **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 $\frac{2BLA2^{2(1+p-\alpha)}}{2^{1+p-\alpha}-1} T^{\frac{1+p-\alpha}{1+p}} + 4F_{\max}\beta_2$.

Proof: At any time t for the set of active hypercubes $\boldsymbol{C}_i(t)$ that the context vector of i belongs to, $l_{\max}(\boldsymbol{C}_i(t))$ is at least the level of the active hypercube $x^d_i(t) \in C^d_i(t)$ for some type-d context. Since a near optimal arm's one-step regret at time t is upper bounded by $2BL2^{-l_{\max}(\boldsymbol{C}_i(t))\alpha}$, the total regret due to near optimal arms by time T is upper bounded by

$$2BL \sum_{t=1}^{T} 2^{-l_{\max}(\boldsymbol{C}_i(t))\alpha} \leq 2BL \sum_{t=1}^{T} 2^{-l(C^d_i(t))\alpha}.$$

Let $l_{\max,u}$ be the maximum level type-d hypercube when type-d contexts are uniformly distributed by time T. We must have

$$A \sum_{l=1}^{l_{\max,u}-1} 2^l 2^{pl} < T \tag{4}$$

otherwise the highest level hypercube by time T will be $l_{\max,u} - 1$. Solving (4) for $l_{\max,u}$, we get $l_{\max,u} < 1 + \log_2(T)/(1+p)$. $\sum_{t=1}^{T} 2^{-l(C^d_i(t))\alpha}$ takes its greatest value when type-d context arrivals by time T is uniformly distributed in $\mathcal{X}_d$. Therefore we have

$$\sum_{t=1}^{T} 2^{-l(C^d_i(t))\alpha} \leq \sum_{l=0}^{l_{\max,u}} 2^l A 2^{pl} 2^{-\alpha l} < \frac{A2^{2(1+p-\alpha)}}{2^{1+p-\alpha}-1} T^{\frac{1+p-\alpha}{1+p}}.$$

∎

From Lemma 6, 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 2, 3, 4 and 6 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

$$R_i(T) \leq T^{f_1(\alpha)} \left(8DZ_i \log T + \frac{2BLA2^{2+\alpha+\sqrt{9\alpha^2+8\alpha}}}{2^{\frac{2+\alpha+\sqrt{9\alpha^2+8\alpha}}{2}} - 1} \right)$$
$$+ T^{f_2(\alpha)} 8(M-1)F_{\max}\beta_2(3\alpha + \sqrt{9\alpha^2+8\alpha})/(4\alpha)$$
$$+ T^{f_3(\alpha)}(8DZ_i + 4(M-1)F_{\max}\beta_2) + 2(1+D)|\mathcal{F}_i|\beta_2,$$

where $Z_i = |\mathcal{F}_i| + (M-1)(F_{\max}+1)$, $f_1(\alpha) = (2 + \alpha + \sqrt{9\alpha^2+8\alpha})/(2 + 3\alpha + \sqrt{9\alpha^2+8\alpha})$, $f_2(\alpha) = 2\alpha/(3\alpha + \sqrt{9\alpha^2+8\alpha})$, $f_3(\alpha) = 2/(2 + 3\alpha + \sqrt{9\alpha^2+8\alpha})$.

Proof: For each hypercube of each type-d context, the regret due to trainings and explorations is bounded by Lemma 2. It can be shown that for each type-d context there can be at most $4T^{1/(1+p)}$ hypercubes that is activated by time T. Using this we get a $O(T^{z+1/(1+p)} \log T)$ upper bound on the regret due to explorations and trainings for a type-d context. Then we sum over all types of contexts $d \in \mathcal{D}$. We show in Lemma 6 that the regret due to near optimal arm selections in exploitation steps is $O(T^{\frac{1+p-\alpha}{1+p}})$. In order to balance the time order of regret due to explorations, trainings and near optimal arm selections in exploitations, while at the same time minimizing the number of explorations and trainings, we set $z = 2\alpha/p$, and $p = \frac{3\alpha+\sqrt{9\alpha^2+8\alpha}}{2}$, which is the value which balances these two terms. Notice that we do not need to balance the order of regret due to suboptimal arm selections since its order is always less than the order of trainings and explorations. We get the final result by summing these two terms together with the regret due to suboptimal arm selections in exploitation steps which is given in Lemma 3. ∎

From the result of Theorem 1, it is seen 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 $F_{\max}$. Another observation is that the time order of the regret does not depend on the number of types of contexts, i.e., D. Therefore, the regret bound we have in this paper and its analysis is significantly different from the regret bounds we had in our prior work [30] for algorithms which do not adaptively choose the type of the context to exploit, whose time order approaches linear as D increases. More is discussed about this in Section VI. Moreover, the regret bound in Theorem 1 is a worst-case bound which holds for any distribution over the data, label and context space satisfying the condition in Definition 1. Depending on this distribution, the number of trainings may be much smaller. However, this will not change the time order of the regret but it will only change the time-independent constants that multiplies the time order.

C. A note on performance of ACAP for a single learner system

Although ACAP is developed such that multiple learners can cooperate to receive higher rewards than the case they do not cooperate, ACAP can also be used when there is only one learner in the system with set of classifiers $\mathcal{F}_i$. Then the action set of the learner will be $\mathcal{K}_i = \mathcal{F}_i$, and the training phase is no more needed since there are no other learners. An analysis similar to the one in the previous subsection can be carried out for ACAP with a single learner. This will give a result such that the time order of the regret will be the same as in Theorem 1, while the constants that multiply the time order will be much smaller. Basically, instead of the multiplicative constants of the form $MF_{\max}$, we will have multiplicative constants of the form $|\mathcal{F}_i|$. Detailed numerical analysis of ACAP for a single learner system is given in Section VII.

V. Adaptive Contexts with Adaptive Partition For Concept Drift

When there is concept drift, accuracies of the classifiers can change over time even for the same context. Because of this the best context to exploit when the context vector is $\boldsymbol{x}$ at time t, can be different from the best context to exploit when the context vector is $\boldsymbol{x}$ at another time t'. ACAP should be modified to take this into account. In this section we assume that the concept drift is gradual as given in Definition 2. ACAP can be modified in the following way to deal with gradual concept drift. The idea is to use a time window of recent observations to calculate the estimated rewards of the actions in $\mathcal{K}_i$ and only use these observations when deciding to train, explore or exploit. We call the modified algorithm ACAP with time window (ACAP-W). This algorithm groups the time slots into rounds $\rho = 1, 2, \ldots$ each having a fixed length of $2\tau_h$ time slots, where τ_h is an integer called *the half window length*. The idea is to keep separate control functions and counters for each round, and calculate the estimated accuracies for each type of context in a round based only on the observations that are made during the time window of that round. The control functions for the initialization round of ACAP-W is the same as the control functions $D_1(t)$, $D_2(t)$ and $D_3(t)$ of ACAP, while the control functions for rounds $\rho > 1$ are $D_1^{\tau_h}(t) = D_3^{\tau_h}(t) = (t \mod \tau_h + 1)^z \log(t \mod \tau_h + 1)$ and $D_2^{\tau_h}(t) = F_{\max}(t \mod \tau_h + 1)^z \log(t \mod \tau_h + 1)$, for some $0 < z < 1$. Each round ρ is divided into two sub-rounds. Except the initialization round, i.e., $\rho = 1$, the first sub-round is called the *passive* sub-round, while the second sub-round is called the *active* sub-round. For the initialization round both sub-rounds are active sub-rounds. In order to reduce the number of trainings and explorations, ACAP-W has an overlapping round structure as shown in Fig. 6. For each round except the initialization round, passive sub-rounds of round ρ, overlaps with the active sub-round of round $\rho-1$. ACAP-W operates in the same way as ACAP in each round. Basically it starts with the entire context space $\mathcal{X}_d$ as a single hypercube for each type $d \in \mathcal{D}$ of context, and adaptively divides the context space into smaller hypercubes the same as as ACAP. We can view ACAP-W as running different instances of ACAP at each round. Let the instance of ACAP that is run by ACAP-W at round ρ be ACAP$_\rho$.

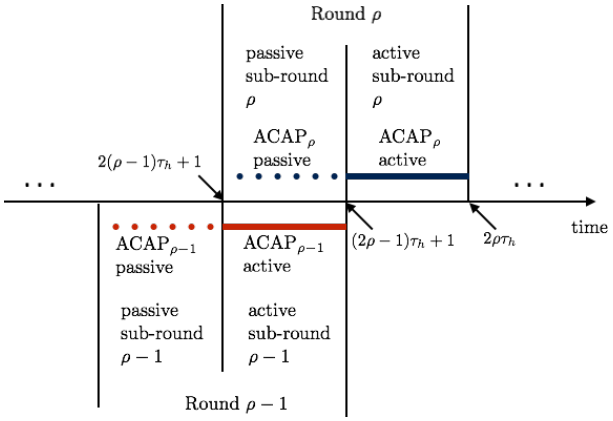

Fig. 6. Operation of ACAP-W showing the round structure and the different instances of ACAP running for each round.

Hypercubes of ACAP$_\rho$ are generated only based on the context observations in round ρ. If time t is in the active sub-round of round ρ, action of learner $i \in \mathcal{M}$ is taken according to ACAP$_\rho$. As a result of this action, sample mean rewards, counters and hypercubes of both ACAP$_\rho$ and ACAP$_{\rho+1}$ are updated. Else if time t is in the passive sub-round of round ρ, action of learner $i \in \mathcal{M}$ is taken according to ACAP$_{\rho-1}$ (see Fig. 6). As a result of this action, sample mean rewards, counters and hypercubes of both ACAP$_{\rho-1}$ and ACAP$_\rho$ are updated. At the start of a round ρ, the accuracy estimates and counters of that round is equal to zero. However, due to the two sub-round structure, when the active sub-round of round ρ starts, learner i already has some observations for the context and actions taken in the passive sub-round of that round, hence depending on the arrivals and actions in the passive sub-round, the learner may even start the active sub-round by exploiting, whereas it should have always spent some time in training and exploration if it starts an active sub-round without any past observations.

In Section IV, learner i's regret was sublinear in T because at any time $t \leq T$, it had at least $O(t^{2\alpha/p} \log t)$ observations for contexts similar to the context at time t, so that in an exploitation step it guarantees to choose the optimal action with a very high probability, where value of p is given in Theorem 1. However, in this section due to the concept drift, contexts that are similar to the context at time t can only come from observations in the current

round. Since round length is fixed, it is impossible to have sublinear number of similar context observations for every t. Thus, achieving sublinear regret under concept drift is not possible. Therefore, in this section we focus on the average regret which is given by $R_i^{\text{avg}}(T) := (1/T)\sum_{t=1}^{T}\left(\pi_{k_i^*(\boldsymbol{x}_i(t),t),t}(\boldsymbol{x}_i(t)) - c^i_{k_i^*(\boldsymbol{x}_i(t),t)}\right) - (1/T)E\left[\sum_{t=1}^{T}(I(\hat{y}_t^i(\alpha_t(\boldsymbol{x}_i(t))) = y_i(t)) - c^i_{\alpha_t(\boldsymbol{x}_i(t))})\right]$, The following theorem gives the average regret of ACAP-W.

.

Theorem 2: When ACAP-W is run with parameters $p = \frac{3\alpha+\sqrt{9\alpha^2+16\alpha}}{2}$, $z = 2\alpha/p$, $D_1^{\tau_h}(t) = D_3^{\tau_h}(t) = (t \mod \tau_h + 1)^z \log(t \mod \tau_h + 1)$, $D_2^{\tau_h}(t) = F_{\max}(t \mod \tau_h + 1)^z \log(t \mod \tau_h + 1)$, and $\tau_h = \lfloor \tau^{(3\alpha+1)/(3\alpha+2)} \rfloor$[9], where τ is the stability of the concept which is given in Definition 2, the average regret of learner i by time T is $R_i^{\text{avg}}(T) = \tilde{O}\left(\tau^{-2\alpha/(4+3\alpha+\sqrt{9\alpha^2+16\alpha})}\right)$, for any $T > 0$.

Proof: (Sketch) The basic idea is to choose τ_h smartly such that the regret due to variation of expected accuracies of classifiers and the regret due to variation of estimated accuracies of classifiers due to limited number of observations during a time window is balanced. The optimal value of $\tau_h = \lfloor \tau^{(3\alpha+1)/(3\alpha+2)} \rfloor$ can be found by doing an analysis similar to the analysis in the proof of Theorem 1 of [31] which is done for finding the optimal uniform partition of the entire context space. The difference here is that partition over time is uniform while partition over each type-d context is adaptive and independent of each other for different types of contexts. In [31], it is shown that using a uniform partition is at least as good as using an adaptive partition when the realized context arrivals are uniformly spaced over the entire space. Therefore using a uniform partition of time will perform at least as good as using an adaptive partition for time. Now consider a variation of ACAP in which we run a new instance of ACAP for rounds of length τ, where both the contexts and time is adaptively partitioned. Basically define a new context vector $\boldsymbol{z} = (z_1, \ldots, z_D)$, where $z_d = (x_d, (t \mod \tau + 1)/\tau)$, for $d \in \mathcal{D}$. We call z_d a type-d pair of contexts. The idea is to adaptively create two-dimensional hypercubes for each type-d pairs of contexts, and exploit according to estimated best type of pairs of contexts. An analysis similar to the analysis of ACAP can be done for this modification, with the control functions, p and z given as in the statement of this theorem, which will give a regret bound of $\tilde{O}\left(\tau^{(4+\alpha+\sqrt{9\alpha^2+16\alpha})/(4+3\alpha+\sqrt{9\alpha^2+16\alpha})}\right)$, for each round of τ time slots (not for τ_h which is for the window length). Since uniform partition of time is better than adaptive partition of time, this will also be an upper bound on the regret of ACAP-W for τ rounds. We get the result for time average regret with dividing this by τ. ∎

From the result of this theorem we see that the average regret decays as the stability of the concept τ increases, and the decay rate is independent of D. This is because, ACAP-W will use a longer time window (round) when τ is large, and thus can get more observations to estimate the sample mean rewards of actions in that round, which will result in better estimates hence smaller number of suboptimal action selections. Another reason is that the average number of trainings and explorations required decrease with the round length.

[9]For a number b, $\lfloor b \rfloor$ denotes the largest integer that is smaller than or equal to b.

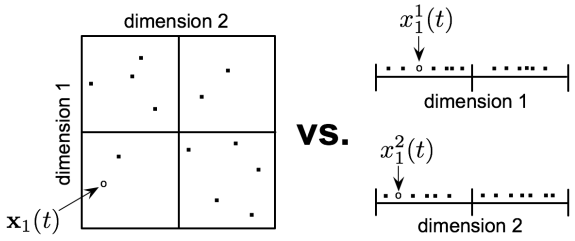

Fig. 7. Curse of dimensionality example with $D = 2$. Although contexts arrived by t are sparse in $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, they are dense in $\mathcal{X}_1$ and $\mathcal{X}_2$.

VI. Comparison with Work in Contextual Bandits

As we mentioned in the introduction section, we have considered the distributed online learning problem with non-adaptive contexts previously [26], [30], [31]. The algorithms proposed in these papers perform well when the dimension of the context vector is small. This is the case either when the types of meta-data a learner can exploit is scarce or each learner has a priori knowledge (e.g., expert knowledge) about which meta-data is the most relevant. However, neither of these assumptions hold in most of the Big Data systems. Therefore, in order to learn which classifiers yield the highest number of correct predictions as fast as possible, learners need to learn which contexts they should take into account when choosing actions. Differences of this work from our previous work are the following. First of all, the hypercubes in DCZA in [31] are formed over the entire context space $\mathcal{X}$, while ACAP forms hypercubes over each $\mathcal{X}_d$ separately. This allows ACAP to differentiate between arrival rates of different types of contexts, thus it can perform type specific training, exploration or exploitation. Moreover, ACAP will not suffer from the curse of dimensionality (context vector arrivals to $\mathcal{X}$ may be sparse even when projections of these context vectors to each dimension is not) as given in the example in Fig. 7, while DCZA will suffer from it. Secondly, although ACAP exploits according to the estimated most relevant context among all the contexts at time t, in all three learning phases, it updates the estimated accuracies of the chosen action for all contexts. Therefore, learning is done in parallel for different types of contexts. Thirdly, when calculating the regret, the optimal we compare against to is different for DCZA and ACAP. Given a context vector $\boldsymbol{x}$, the optimal action for learner i according to the entire context vector is $k^*_{i,\text{DCZA}}(\boldsymbol{x}) = \arg\max_{k\in\mathcal{K}_i} \pi_k(\boldsymbol{x}) - c^i_k$, while the optimal action according to the best type of context is $k_i^*(\boldsymbol{x}) = \arg\max_{k\in\mathcal{K}_i}\left(\max_{x^d\in\boldsymbol{x}} \pi_k^d(x^d) - c^i_k\right), \quad \forall \boldsymbol{x} \in \mathcal{X}$. The following corollary gives a sufficient condition under which these optimal actions coincide.

Corollary 1: When $\max_{x^d\in\boldsymbol{x}} \pi_k^d(x^d) = \pi_k(\boldsymbol{x})$ for all $x^d \in \mathcal{X}_d$, $d \in \mathcal{D}$, we have $k_i^*(\boldsymbol{x}) = k^*_{i,\text{DCZA}}(\boldsymbol{x})$. Otherwise, we have $\pi_{k_i^*(\boldsymbol{x})}(\boldsymbol{x}) \leq \pi_{k^*_{i,\text{DCZA}}(\boldsymbol{x})}(\boldsymbol{x})$. In other words, the optimal policy given the entire context vector is at least as good as the optimal policy given the best type of context.

When the condition given in Corollary 1 does not hold, the regret of ACAP with respect to $k^*_{i,\text{DCZA}}(\boldsymbol{x})$ may increase linearly in time. Therefore, if we are only interested in the asymptotic performance, it is better to use DCZA in that case. However, due to requiring smaller number of trainings and explorations, the performance of ACAP can be better than DCZA for short time horizons. When the condition in Corollary 1 holds, using ACAP, learner i can achieve $\tilde{O}\left(T^{(2+\alpha+\sqrt{9\alpha^2+8\alpha})/(2+3\alpha+\sqrt{9\alpha^2+8\alpha})}\right)$ regret with respect to the optimal policy, while DCZA will achieve $\tilde{O}\left(T^{(2D+\alpha+\sqrt{9\alpha^2+8\alpha D})/(2D+3\alpha+\sqrt{9\alpha^2+8\alpha D})}\right)$ regret with respect to the optimal policy which is worse for $D > 1$. This is due to the fact that DCZA needs to train and explore in a much larger number of hypercubes than ACAP since it creates hypercubes over the entire context space. The worst case number of activated hypercubes in DCZA is $O(T^{D/(D+p)})$ while it is $O(T^{1/(1+p)})$ for ACAP. Whenever the total number of arrivals into an active hypercube exceeds the threshold value, DCZA splits that hypercube into 2^D child hypercubes, while ACAP only splits it into 2 child hypercubes. In summary, ACAP is not affected from the curse of dimensionality because it treats each type of context separately. CLUP in [31] is similar to DCZA but it uses a non-adaptive uniform partition of the entire context space.

The idea of adaptive partitioning of the context space is also investigated in [9] but only in a single learner setting. But the method of partitioning the context space is different than DCZA and CLUP. However, the algorithm in [9] also creates an adaptive partition over the entire context space, hence suffers from the curse of dimensionality.

VII. Numerical Results

In this section, we numerically compare the performance of our learning algorithms with state–of–the–art online ensemble learning techniques and with other bandit-type schemes.

A. Data Sets

We consider seven data sets, described below. The first four data sets, **R1**–**R4**, are well known in the data mining community and refer to real–world problems. In particular, the first three data sets are widely used by the literature dealing with concept drift. For a more detailed description of these data sets we refer the reader to the cited references. The last three data sets, **S1**–**S3**, are synthetic data sets.

R1: Network Intrusion [28], [29], [33], [34]. The network intrusion data set from UCI archive [33] consists of a series of TCP connection records, labeled either as normal connections or as attacks.

R2: Electricity Pricing [34]–[36]. 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 [29], [36], [37]. The forest cover type data set from UCI archive [33] 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 [34], [38]. In the credit card risk assessment data set, used for the PAKDD 2009 Data Mining Competition [38], 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.

S1: Piecewise Linear Classifiers. In this data set we consider a 4–dimensional feature space. In each time step t, the data $s_1(t) = \left(s_1^1(t), s_1^2(t), s_1^3(t), s_1^4(t)\right)$ is drawn uniformly from the interval $[-1, 1]^4$. The labels are such that $y_1(t) = 1$ if $s_1^4(t) > \max\left(h_1(s_1^1(t)), h_2(s_1^2(t))\right)$, otherwise $y_1(t) = 0$. Notice that the label is independent from the third feature. $h_1(\cdot)$ and $h_2(\cdot)$ can be generic functions. We consider $h_1(s_1^1(t)) = \frac{1}{10 s_1^1(t)}$ and $h_2(s_1^2(t)) = (s_1^2(t))^2$. We will adopt linear classifiers trained in different intervals of the feature space, in this way the non–linear functions $h_1(\cdot)$ and $h_2(\cdot)$ will be approximated by piecewise linear classifiers [39].

S2: Piecewise Linear Classifiers, abrupt concept drift. This data set is similar to **S1**, the only difference is that in the testing phase we have $h_2(s_1^2(t)) = w_2 \cdot (s_1^2(t))^2$. The weight w_2 is set to 1 for the first half of the testing set, and then it is set to 0, i.e., in the second part of the testing set the label is independent from the second feature.

S3: Piecewise Linear Classifiers, gradual concept drift. Similarly to **S2**, we consider $h_2(s_1^2(t)) = w_2 \cdot (s_1^2(t))^2$ and we initialize $w_2 = 1$, but in this case we gradually decrease w_2 at each step by an amount $\frac{1}{N_{test}}$, where N_{test} represents the number of instances in the testing set.

B. Comparison with ensemble schemes

In this subsection we compare the performance of our learning algorithms with state–of–the–art online ensemble learning techniques, listed in Table II. Different from our algorithms 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 [46]. 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). 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**, **R3**, **S1**, **S2**, **S3**, and $2,000$ for **R2** and **R4**. For **S1**, **S2**, and **S3**, each classifier c of the first 4 classifiers is trained for data such that $s_1^1(t) \in [\frac{c-1}{4}, \frac{c}{4}]$. whereas each classifier c of the last 4 classifiers is trained for data such that $s_1^2(t) \in [\frac{c-1}{4}, \frac{c}{4}]$. In this way each classifier is trained to predict accurately a specific interval of the feature space. Then we take other sequential records ($20,000$ for **R1**, **R3**, **S1**, **S2**, **S3**, and $8,000$ for **R2** and **R4**) to generate a set in which the local classifiers are tested, and the results are used to train Adaboost. Finally, we

Abbreviation	Name of the Scheme	Reference	Performance						
			R1	R2	R3	R4	S1	S2	S3
AM	Average Majority	[28]	3.07	41.8	29.5	34.1	35.4	27.7	25.5
Ada	Adaboost	[40]	3.07	41.8	29.5	34.1	35.4	27.7	25.5
OnAda	Fan's Online Adaboost	[41]	2.25	41.9	39.3	19.8	32.7	27.1	26.2
Wang	Wang's Online Adaboost	[42]	1.73	40.5	32.7	19.8	17.8	14.3	13.6
DDD	Diversity for Dealing with Drifts	[34]	1.15	44.0	23.9	19.9	43.0	38.0	37.9
WM	Weighted Majority algorithm	[43]	0.29	22.9	14.1	67.4	39.2	30.7	29.5
Blum	Blum's variant of WM	[44]	1.64	40.3	22.6	68.1	39.3	31.7	30.2
TrackExp	Herbster's variant of WM	[45]	0.52	23.0	14.8	22.0	31.9	25.0	23.0
ACAP	**Adaptive Contexts with Adaptive Partition**	our work	**0.71**	**5.8**	**19.2**	**19.9**	**6.9**	**7.2**	**7.9**
ACAP–W	**ACAP with Time Window**	our work	**0.91**	**19.4**	**20.2**	**20.2**	**8.0**	**6.8**	**7.8**

TABLE II
COMPARISON AMONG ACAP AND OTHER ENSEMBLE SCHEMES: PERCENTAGES OF MIS–CLASSIFICATIONS IN THE DATA SETS **R1**–**R4** AND **S1**–**S3**.

select other sequential records ($20,000$ for **R1** and **R3**, **S1**, **S2**, **S3**, $21,000$ for **R2**, and $26,000$ for **R4**) to generate the testing set that is used to run the simulations and test all the considered schemes.

For our schemes (ACAP and ACAP-W) 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. For the data sets **S1**–**S3** the context vector is represented by the first three features. Each context dimension is normalized such that contexts belong to $[0, 1]$.

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**, **S1**, **S2**, and **S3**. ACAP is far more accurate than all the ensemble schemes in the data sets **S1**–**S3**. In these data sets each classifier is expert to predict in specific intervals of the feature space. Our results prove that, in these cases, it is better to choose smartly a single classifier instead of combining the predictions of all the classifiers. Notice that ACAP is very accurate also in presence of abrupt concept drift (**S2**) and in presence of gradual concept drift (**S3**). However, in these cases the sliding window version of our scheme, ACAP-W, performs (slightly) better than ACAP because it is able to adapt quickly to changes in concept.

Now we investigate how ACAP and ACAP-W learn the optimal context for the data sets **S2** and **S3**. Fig. 8 shows the cumulative number of times ACAP and ACAP-W use context 1, 2, and 3 to decide the classifier or the learner to sent the data to. The top–left subfigure of Fig. 8 refers to the data set **S2** and to the decisions made by ACAP. ACAP learns quickly that context 3 is not correlated to the label; in fact, for its decisions it exploits context 3 few times. Until time instant $10,000$ ACAP selects equally among context 1 and 2. This means that half the times the contextual information $x_i^1(t)$ is more relevant than the contextual information $x_i^2(t)$, and in these cases ACAP selects a classifier/learner that is expert

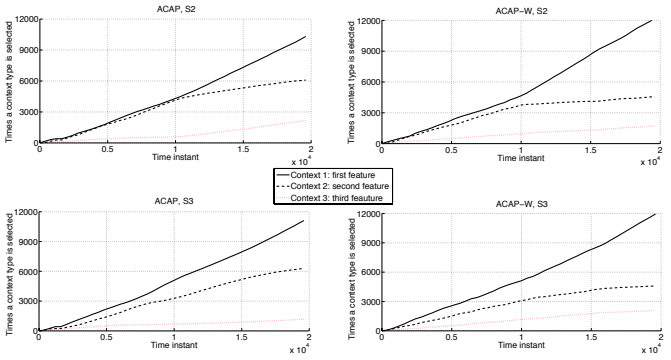

Fig. 8. Cumulative number of times ACAP and ACAP-W use context 1, 2, and 3 to take decisions, for the datasets **S2** and **S3**.

Abbreviation	Reference	Performance		
		S1	S2	S3
UCB1	[8]	18.3	20.0	23.8
Adap1	[31]	9.6	11.4	9.2
Adap2	[31]	11.8	19.1	17.4
Adap3	[31]	19.9	23.5	20.0
ACAP	our work	**7.4**	**7.7**	**7.9**
ACAP–W	our work	**9.1**	**7.5**	**7.6**

TABLE III
COMPARISON AMONG ACAP AND OTHER BANDIT–TYPE SCHEMES: PERCENTAGES OF MIS–CLASSIFICATIONS IN THE DATA SETS **S1**–**S3**.

to predict data with contextual information similar to $x_i^1(t)$ (notice that such classifier/learner is automatically learnt by ACAP). At time instant $10,000$ an abrupt drift happens and context 2 becomes suddenly irrelevant, as context 3. ACAP automatically adapts to this situation, decreasing the number of times context 2 is exploited. However, this adaptation is slow because of the large sample mean reward obtained by context 2 during the first part of the simulation. ACAP-W, whose decisions for data set **S2** are depicted in the top–right subfigure of Fig. 8, helps to deal with this issue. In fact, the rewards obtained in the past are set to 0 at the beginning of a new window, and the scheme is able to adapt quickly to the abrupt drift. The bottom–left and bottom–right subfigures of Fig. 8 refer to the decisions made by ACAP and ACAP-W, respectively, for data set **S3**, which is affected by gradual drift. Both ACAP and ACAP-W adapt gradually to the concept drift, but also in this scenario ACAP-W adapts faster than ACAP.

C. Comparison with bandit-type schemes

In this subsection we compare the performance of our learning algorithms with other bandit–type schemes, listed in Table II. UCB1 [8] is a bandit scheme which does not exploit

the contextual information, hence it must learn the best single classifier to exploit. Adap1, Adap2, and Adap3 are the DCZA scheme proposed in [31], they use only the first, second, and third dimensions of the context, respectively, instead of using all three dimensions at the same time. We used the same training and testing procedures described in Subsection VII-B, but we consider a single learner scenario (i.e., a unique learner possesses all the classifiers).

The results show that the bandit schemes that adopt adaptive contextual information are more accurate than UCB1 that does not exploit the context. However, it is extremely important to learn the right context to use in different situations. In fact, the accuracy of Adap3 is very low because context 3 is irrelevant. Moreover, the performance of Adap1 and Adap2, that exploit relevant contexts, are not as good as the performance of our schemes. In fact, ACAP and ACAP-W are able to learn the best contexts to exploit for each possible context vector and, in addition to it, since they update the sample mean reward of all the contexts in parallel, they learn as fast as a bandit scheme that uses only a single dimension of the context vector.

VIII. Conclusion

In this paper we considered the problem of adaptively learning which contexts to exploit in Big Data stream mining. We proposed a novel learning framework and answered the following questions: Does using the context information help improve the performance of the online stream mining system? What is the best way to choose which context to consider when making decisions? How should the learning process be performed when there is concept drift? 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] S. Avidan, "Support vector tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 8, pp. 1064–1072, 2004.

[5] 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.

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] M. Sewell, "Ensemble learning," *RN*, vol. 11, no. 02, 2008.

[21] E. Alpaydin, *Introduction to machine learning*. The MIT Press, 2004.

[22] S. McConnell and D. B. Skillicorn, "Building predictors from vertically distributed data," in *Proc. of the 2004 conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 2004, pp. 150–162.

[23] 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.

[24] 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.

[25] 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.

[26] C. Tekin and M. van der Schaar, "Decentralized online big data classification - a bandit framework," *arXiv preprint arXiv:1308.4565*, 2013.

[27] I. Zliobaite, "Learning under concept drift: an overview," Vilnius University, Faculty of Mathematics and Informatics, Tech. Rep., 2010. [Online]. Available: http://arxiv.org/abs/1010.4784

[28] 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.

[29] 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.

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

[31] ——, "Distributed online learning via cooperative contextual bandits," *submitted to Signal Processing, IEEE Transactions on. arXiv preprint arXiv:1308.4568*, 2013.

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

[33] 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.

[34] 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.

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

[36] 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.

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

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

[39] J. Sklansky and L. Michelotti, "Locally trained piecewise linear classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 2, pp. 101–111, 1980.
[40] 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.
[41] 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.
[42] 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.
[43] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.
[44] 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.
[45] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, Aug. 1998.
[46] D. S. Rosario, "Highly effective logistic regression model for signal (anomaly) detection," in *Proc. IEEE ICASSP*, vol. 5, 2004, pp. 817–820.