Online Big Data Stream Mining Using Adaptive Contexts

Cem Tekin * Mihaela van der Schaar †

Abstract

A plethora of stream mining applications have emerged which require classification of large data streams generated by distributed and heterogeneous sources. This data is processed by decentralized learners who have different sets of classification functions to make predictions about the data. However, the accuracy of the predictions of a classification function is unknown initially to the learner, its data dependent and time varying. Besides these unknowns, stream classification has high computation and delay costs due to the high dimensionality of the data. Each learner, if acting independently, makes predictions solely based on the limited number of classification functions it has and limited data streams it processes. Therefore, it is important for the learner to choose the best classification function for each specific data stream. However, a learner can also get help from other learners when making predictions about its own data stream. The data streams are characterized by their context information which can be used as meta-data to choose which type of classifiers should be deployed to make better predictions. In this paper, we propose a context-adaptive learning algorithm which learns online what is the best context to deploy to select what learners and classification functions 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. We theoretically bound the performance gap between our context-adaptive learning algorithm and a benchmark policy that knows everything about classification accuracies, data and context arrivals. Our numerical results illustrate that our algorithm outperforms most of 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.

1 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 distributed network and traffic monitors, multimedia sources, sensor networks, etc. The input data stream is heterogeneous and dynamically evolves over time. Automatically generated meta-data is often associated with these data streams, i.e. contexts that represent any information related to the input data stream such as time, location and type (e.g., data features/characteristics/modality) information, which is available to the learners before the actual classification and can be used to select which classification function is used to make predictions. Our prior work has shown that the performance of existing learning algorithms usually depends on the dimension of the context space [5]. In general, the performance degrades exponentially with the dimension of the context space which makes it impractical to exploit all the context information available to the learners. Instead, learners need to choose which contexts they should base their decisions on. This is because contexts may have different types of correlations with the data stream itself, which results in different accuracies depending on which classification function is chosen. Each learner can process (label) the incoming data in two different ways: either it can exploit its own information and its own classification functions 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 classification functions 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 slot. 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. 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 a distributed data mining system. We consider cooperative learners which classify other's data when requested, but instead of maximizing the system

*University of California, Los Angeles.
†University of California, Los Angeles.

utility function, a learner's goal is to maximize its individual utility. However, it can be shown that when the classification costs capture the cost to the learner which is cooperating with another learner to classify its data, maximizing the individual utility corresponds to maximizing the system utility.

To jointly optimize the performance of the distributed data mining system, we design distributed online learning algorithms whose long-term average rewards converge to the best distributed solution which can be obtained for the classification problem given complete knowledge of online data characteristics as well as their classification function accuracies and costs when applied to this data. We extend the novel cooperative contextual bandit framework in [5] to learn the best contexts to exploit adaptively over time, and show that this significantly improves the performance of online big data mining systems. The benchmark we compare against is a genie aided scheme, in which the genie knows classification accuracies of each classification function of each learner, and chooses the classification function which yields the highest expected accuracy for the best context in the set of available contexts at each time step. We call the difference between the expected total reward (correct predictions minus cost) of learner i under the genie aided scheme and the expected total reward of the online learning algorithm used by learner i as the regret of learner i, and show that it is sublinear in time. The time order is independent of the dimension of the context space, in contrast to [5], where the time order is linear in the dimension of the context space.

When time is incorperated in the context, we show that our distributed contextual learning framework can be used to deal with *concept drift* [6], which occurs when the distribution of problem instances changes over time. To illustrate the advantage of our approach which adaptively chooses among different contexts to exploit at each time step, we provide numerical results by applying our learning algorithms to the classification of various data streams and compare the results with existing state-of-the-art solutions. Although in this paper we focus on a distributed multi-learner data mining system, the idea of context-adaptive learning is also new for a single learner data mining system. Our algorithm and results can be easily modified to work in single learner system.

2 Related Work

Online learning in distributed data classification systems aims to address the informational decentralization, communication costs and privacy issues arising in these systems. In these systems, learning rates are different because either each learner observes the entire feature space but has access to a subset of instances of the entire data set, which is called *horizontally distributed* data, or each learner has access to only a subset of the features but the instances can come from the entire data set, which is called *vertically distributed* data. For example in [7–10], various solutions are proposed for distributed data mining problems of horizontally distributed data, while in [11, 12], ensemble learning techniques are developed that exploit the correlation between the local learners for vertically distributed data. Several cooperative distributed data mining techniques are proposed in [12–15], where the goal is to improve the prediction accuracy with costly communication between local predictors. 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 with the same context information. This novel approach allows us to deal with both horizontally and vertically distributed data in a unified manner within a distributed data mining system. Although our framework and illustrative results are depicted using horizontally distributed data, if context is changed to be the set of relevant features, then our framework and results can operate on vertically distributed data. Moreover, we assume no prior knowledge of the data and context arrival processes and classification function accuracies, and the learning is done in a non-Bayesian way. Learning in a non-Bayesian way is appropriate in decentralized system since learners often do not have correct beliefs about the distributed system dynamics.

Most of the prior work in distributed data 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 step with respect to the optimal solution. In other words, we prove regret bounds that hold uniformly over time. Some of the existing solutions (including [9, 10, 16–21]) propose ensemble learning techniques. In our work we only consider choosing the best classification function (initially unknown) from a set of classification functions that are accessible by decentralized learners. However, our proposed distributed learning methods can easily be adapted to perform ensemble learning. We provide a detailed comparison to our work in Table 1.

Other than distributed data mining, our learning framework can be applied to any problem that can be formulated as a decentralized contextual bandit problem [5]. Contextual bandits have been studied before in [22, 23] and other works in a single agent setting. Different from these work we consider decentralized

	[9, 14, 19–21]	[13, 15]	[11]	This work
Aggregation	non-cooperative	cooperative	cooperative	no
Message exchange	none	data	training residual	data and label only if improves performance
Learning approach	offline/ online	offline	offline	Non-bayesian online
Correlation exploitation	N/A	no	no	yes
Information from other learners	no	all	all	only if improves accuracy
Data partition	horizontal	horizontal	vertical	horizontal and vertical
Bound on regret	no	no	no	sublinear
Learning the best context	no	no	no	yes

Table 1: Comparison with related work in distributed data mining.

agents and learning the best type of context to exploit adaptively over time.

3 Problem Formulation

The system model is shown in Fig. 1. There are M learners which are indexed by the set $\mathcal{M} := \{1, 2, \ldots, M\}$. Let $\mathcal{M}_{-i} := \mathcal{M} - \{i\}$ be the set of learners learner i can choose from to send its data for classification. These learners work in a discrete time setting $t = 1, 2, \ldots, T$, where the following events happen sequentially, in each time slot: (i) a data stream $s_i(t)$ 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 $x_i^d(t) \in \mathcal{X}_d$ for $d \in \mathcal{D} := \{1, \ldots, D\}$, where $\mathcal{X}_d$ is the set of type-d contexts[1], (ii) each learner chooses one of its own classification functions or another learner to send its data and context, and produces a label based on the prediction of its own classification function or the learner to which it sent its data and context, (iii) the truth (true label) is revealed eventually, perhaps by events or by a supervisor, only to the learner 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.

Each learner $i \in \mathcal{M}$ has access to a set of classification functions $\mathcal{F}_i$ which it can invoke to classify the data. The performance of these classification functions are data dependent and unknown a priori. Also the learners have no prior beliefs about the accuracy of these classification functions. Given that the context is $x^d \in \mathcal{X}_d$, the prediction of classification function $f \in \mathcal{F}$ is a Bernoulli random variable with expectation $\pi_f^d(x^d)$. This expectation is defined in the following way. The data stream $s_i(t)$ that comes with context $\boldsymbol{x} \in \mathcal{X} := \mathcal{X}_1 \times \ldots \times \mathcal{X}_D$ is drawn from an unknown but fixed distribution $F(\boldsymbol{x})$ which is a function of the context. This distribution is a fixed function of the context, but when one of the type-d contexts is time[2], the distribution will be time varying. Let $\pi_f(\boldsymbol{x})$ of $f \in \mathcal{F}$ be the expected accuracy of classification function f, where the expectation is taken with respect to distribution $F(\boldsymbol{x})$. Let $\boldsymbol{x}^{-d} := (x^1, \ldots, x^{d-1}, x^{d+1}, \ldots, x^D)$. We define $\pi_f^d(x^d)$ as $\pi_f^d(x^d) := \int_{\boldsymbol{x}^{-d}} \pi_f(\boldsymbol{x}) dF(\boldsymbol{x})$. In our problem, the unknowns for learner i are (i) $\mathcal{F}_j$, $j \in \mathcal{M}_{-i}$, (ii) $F(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$, (iii) $\pi_f(\boldsymbol{x})$, $f \in \mathcal{F}_i$, $\boldsymbol{x} \in \mathcal{X}$, (iv) $\pi_f^d(x^d)$, $f \in \mathcal{F}_i$, $x^d \in \mathcal{X}_d$, $d \in 1, \ldots, D$. 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 classification functions that each learner has, i.e., $F_{\max}$ on $|\mathcal{F}_{j_i}|$[4], $j_i \in \mathcal{M}_{-i}$. Let $\mathcal{K}_i := \mathcal{F}_i \cup \mathcal{M}_{-i}$. We call $\mathcal{K}_i$ the set of arms (alternatives) for learner i. We use index k to denote any arm in $\mathcal{K}_i$, k_i to denote the set classification functions of i, i.e., the elements of the set $\mathcal{F}_i$, j_i to denote other learners in $\mathcal{M}_{-i}$. Let $\mathcal{F} := \cup_{j \in \mathcal{M}} \mathcal{F}_j$ denote the set of all arms of all learners. We use index f to denote an element of $\mathcal{F}$.

At each time step t, learner i can either invoke one of its classification functions or forward the data to another learner to have it labeled. We assume that for learner i, calling each classification function $k_i \in \mathcal{F}_i$ incurs a cost $c_{k_i}^i$. 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 classification function. We assume that a learner can only call a single function for each input data in order to label it. This is a reasonable assumption when the application is delay sensitive since calling more than one classification function increases the delay. 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 a data stream is sent to another learner $j_i \in \mathcal{M}_{-i}$ a cost of $c_{j_i}^i$ is incurred by learner i[5]. Since the costs are bounded, without loss of generality we assume that costs are normalized, i.e., $c_k^i \in [0, 1]$ for all $k \in \mathcal{K}_i$. The learners are cooperative which implies that learner $j_i \in \mathcal{M}_{-i}$ will

[1] We do not make any assumptions about how the context is generated. It can be non-stochastic as well as stochastic.

[2] For example, if the final time is T, one of the contexts can be the normalized time $t_N = t/T$, $t = 1, \ldots, T$.

[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 classification function chosen by learner j_i. Since the learners are cooperative, j_i will obey the rules of the proposed algorithm when choosing a classification function to label i's data.

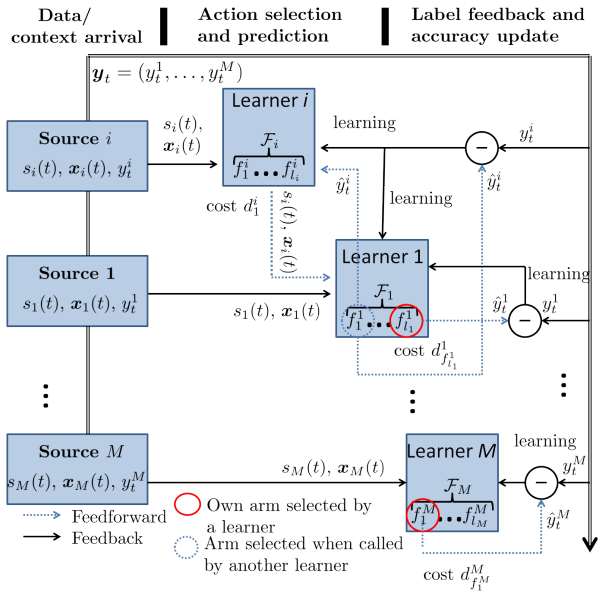

Figure 1: Operation of the distributed data classification system during a time slot.

return a label to i when called by i. 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, however, since our results hold for the case when other learners are not helping i to learn about its own classification functions, they will hold when other learners help i to learn about its own classification functions. If we assume that $c_{j_i}^i$ also captures the cost to learner j_i to classify and send the label back to learner i, then maximizing i's own utility corresponds to maximizing the system utility.

We assume that each classification function produces a binary label[6]. The data stream at time t arrives to learner i with context information $\boldsymbol{x}_i(t) = (x_i^1(t), \ldots, x_i^D(t))$. The context may be generated as a result of pre-classification or a header of the data stream. For simplicity we assume that for each type-d context $\mathcal{X}_d = [0,1]$, while our results can simply be generalized to any bounded interval $[a,b]$. 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 classification function, i.e., $\pi_{j_i}^d(x^d) = \max_{k_{j_i} \in \mathcal{F}_{j_i}} \pi_{k_{j_i}}^d(x^d)$.

Different classification functions can have different accuracies for the same context. Although we do not make any assumptions about the classification accuracy $\pi_k^d(x^d)$ and the classification cost c_k^i for $k \in \mathcal{K}_i$, in general one can assume that classification accuracy increases with classification cost (e.g., classification functions with higher resolution, better processing). In this paper the cost c_k^i is a generic term that can represent any known cost such as processing cost, delay cost, communication cost, etc. We assume that each classification function has similar accuracies for similar contexts; we formalize this in terms of a (uniform) Lipschitz condition.

ASSUMPTION 1. *For each $f \in \mathcal{F}$ and $d \in \mathcal{D}$, there exists $L > 0$, $\alpha > 0$ such that for all $x^d, (x')^d \in \mathcal{X}_d$, we have $|\pi_f^d(x^d) - \pi_f^d((x')^d)| \leq L|x^d - (x')^d|^\alpha$.*

Assumption 1 indicates that the accuracy of a classification function for similar contexts will be similar to each other. For example, the context can be the time of the day or/and the location from which the data originates. Therefore, the relation between the classification accuracy and time can be written down as a Lipschitz condition. We assume that α is known by the learners, while L does not need to be known. An unknown α can be estimated online using the sample mean estimates of accuracies for similar contexts, and our proposed algorithm can be modified to include the estimation of α. The goal of learner i is to explore the alternatives in $\mathcal{K}_i$ to learn the accuracies, while at the same time exploiting the best alternative for the context $\boldsymbol{x}_i(t)$ arriving at each time step t that balances the accuracy and cost to minimize its long term loss due to uncertainty. Learner i's problem can be modeled as a contextual bandit problem [22, 23]. After labeling the input at time t, each learner observes the true label and updates the sample mean accuracy of the selected arm based on this. Accuracies translate into rewards in bandit problems. In the next subsections, we define the benchmark solution and the regret which is the performance loss due to uncertainty about classification accuracies.

3.1 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 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 $k_i^*(\boldsymbol{x}) = \arg\max_{k \in \mathcal{K}_i} \left(\max_{x^d \in \boldsymbol{x}} \pi_k^d(x^d) - c_k^i\right)$, $\forall \boldsymbol{x} \in \mathcal{X}$. Knowing the optimal solution means that learner i knows the classification function in $\mathcal{F}$ that yields the highest expected accuracy for each $x^d \in \mathcal{X}_d$, $d \in \mathcal{D}$. Choosing the best classification function 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.

[6] 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.

3.2 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 arm $\alpha_t(\boldsymbol{x}_i(t))$ at time t for learner i is defined with respect to the best arm $k_i^*(\boldsymbol{x}_i(t))$ at time t. The regret of a learning algorithm for learner i is given by $R_i(T) := \sum_{t=1}^{T}\left(\pi_{k_i^*(\boldsymbol{x}_i(t))}(\boldsymbol{x}_i(t)) - c^i_{k_i^*(\boldsymbol{x}_i(t))}\right) - E\left[\sum_{t=1}^{T}(I(\hat{y}_t^i(\alpha_t(\boldsymbol{x}_i(t))) = y_t^i) - c^i_{\alpha_t(\boldsymbol{x}_i(t))})\right]$, where $\hat{y}_t^i(.)$ denotes the prediction of the arm selected by learner i at time t, y_t^i 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.

4 Adaptive Contexts with Adaptive Partition

In this section we propose an online learning algorithm which adaptively explores different types of contexts and their context spaces to maximize the total expected reward of each learner. 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 classification functions. We name our algorithm *adaptive contexts and adaptive partitions* (ACAP).

4.1 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 high 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. ACAP also adaptively chooses the best context over time from the context vector to choose which arm to exploit, by forming sample mean estimates of the expected accuracies of the arms as a function of the contexts. These sample mean accuracies are updated in parallel for each context, 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. While the decision made by ACAP is to choose an arm in $\mathcal{K}_i$ at each time step, this decision is only based on one type of context among contexts in $\mathcal{D}$. 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. An illustration that shows how partitions of different types of contexts may evolve for ACAP is given in Fig. 2. The idea of zooming into the regions of context space with high arrivals is previously addressed in [22] by activating balls with smaller radius over time. However, the results in [22] cannot be easily generalized to a distributed setting. The first reason is that each learner may have different active balls for the same context at the same time, and the centers of these balls depend on the previous context arrivals, therefore the number of active balls may be much higher than the number of active hypercubes which divides the context space in a more efficient way. Secondly, the training and exploration phases will require keeping information about all the active balls of all learners including their center and radius information. Thirdly, balls in [22] are formed over the entire context space $\mathcal{X}_d$, while we form different hypercubes for each $\mathcal{X}_d$. This allows us to differentiate between different types of contexts. For example if type-d context has few arrivals in interval $[a, b]$ and if type-d' context has a lot of arrivals in interval $[a, b]$, then ACAP will estimate the expected accuracy of arms by using past observations from low level hypercubes for type-d context and high level hypercubes for type-d' context, while the algorithm in [22] will use a single ball over $\mathcal{X}_d \times \mathcal{X}_{d'}$. Finally, whenever another learner is selected by a learner, the exact position of the context should be sent to that learner in [22], while in our work only the information about which hypercube context belongs to is enough to make a decision.

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. The pseudocode of ACAP is given in Fig. 3, and the training, exploration, exploitation and initialization modules are given in Fig. 5 and Fig. 4.

For each type-d context, we call an interval $(a2^{-l}, (a+1)2^{-l}] \subset [0, 1]$ a level l hypercube for $a = 1, \ldots, 2^l - 1$[7]. 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 step, ACAP keeps for learner i a set of hypercubes that cover the con-

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

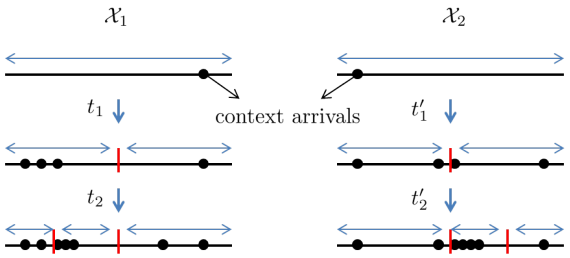

Figure 2: An illustration showing how the adaptive partitions are generated for different types of contexts over time. In this example $D = 2$. The times when a new partition for type-d context is created i.e., t_1, t_2, t_1' and t_2' depends on the context arrival process and can be different for different types of contexts.

text space of each type of context which are mutually exclusive. 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 previous observations and actions taken on $C_i^d(t)$, $d \in \mathcal{D}$. Let $N_C^{i,d}(t)$ be the number of times type-d contexts have arrived to hypercube C of learner i by 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 explore, $D_3(t)$ controls when to explore or exploit. When type-d context is selected as the *main context* at time t, for an arm $k_i \in \mathcal{F}_i$, 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 $j_i \in \mathcal{M}_{-i}$. This is because learner i cannot choose the arm that is used by learner j_i. If the estimated rewards of arms of learner j_i are inaccurate, i's estimate of j_i's reward will be different from the expected reward of j_i's optimal arm. Therefore, learner i uses the rewards from learner j_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 context arrivals to learner j_i in set $C_i^d(t)$ is small, learner i *trains* learner j_i by sending its context to j_i, receiving back the prediction of the classification function chosen by

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

1: Input: $D_1(t)$, $D_2(t)$, $D_3(t)$, p, A
2: 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**($\mathcal{A}_i$)
3: 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, $\boldsymbol{N}_k^i = (N_{k,C^d(t)}^{i,d})_{d \in \mathcal{D}}$, $k \in \mathcal{K}_i$, $\boldsymbol{N}_{1,k}^i = (N_{1,k,C^d(t)}^{i,d})_{d \in \mathcal{D}}$, $k \in \mathcal{M}_{-i}$, $\boldsymbol{N}^i = (\boldsymbol{N}_k^i)_{k \in \mathcal{K}_i}$.
4: **while** $t \geq 1$ **do**
5: **if** $\exists d \in \mathcal{D}$ and $\exists k \in \mathcal{F}_i$ such that $N_{k,C}^{i,d} \leq D_1(t)$ **then**
6: Run **Explore**(t, k, $\boldsymbol{N}_k^i$, $\bar{\boldsymbol{r}}_k^i$)
7: **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**
8: Obtain $N_{C^d(t)}^{k,d}(t)$ from learner k.
9: **if** $N_{C^d(t)}^{k,d}(t) = 0$ **then**
10: Ask k to create hypercube $C^d(t)$ for its type-d context, set $N_{1,k,C^d(t)}^{i,d} = 0$
11: **else**
12: Set $N_{1,k,C^d(t)}^{i,d} = N_{C^d(t)}^{k,d}(t) - N_{k,C^d(t)}^{i,d}$
13: **end if**
14: **if** $N_{1,k,C^d(t)}^{i,d} \leq D_2(t)$ **then**
15: Run **Train**(t, k, $\boldsymbol{N}_{1,k}^i$)
16: **else**
17: Go to line 7
18: **end if**
19: **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**
20: Run **Explore**(t, k, $\boldsymbol{N}_k^i$, $\bar{\boldsymbol{r}}_k^i$)
21: **else**
22: Run **Exploit**(t, $\boldsymbol{N}^i$, $\bar{\boldsymbol{r}}^i$, $\mathcal{K}_i$)
23: **end if**
24: $N_{C^d(t)}^{i,d} = N_{C^d(t)}^{i,d} + 1$
25: **for** $d \in \mathcal{D}$ **do**
26: **if** $N_{C^d(t)}^{i,d} \geq A2^{pl_{C^d(t)}}$ **then**
27: Create 2 level $l_{C_d(t)} + 1$ child hypercubes denoted by $\mathcal{A}_{C^d(t)}$
28: Run **Initialize**($\mathcal{A}_{C^d(t)}$)
29: $\mathcal{A}_i = \mathcal{A}_i \cup \mathcal{A}_{C^d(t)} - C^d(t)$
30: **end if**
31: **end for**
32: $t = t + 1$
33: **end while**

Figure 3: Pseudocode of the ACAP algorithm.

j_i and sending the true label at the end of that time step to j_i so that j_i can compute the estimated accuracy of the classification function it had chosen for i. In order to do this, learner i keeps two counters $N_{1,j_i,C}^{i,d}(t)$ and $N_{2,j_i,C}^{i,d}(t)$ for each $C \in \mathcal{A}_d^i(t)$, which are initially set to 0. At the beginning of each time step for which $N_{1,j_i,C}^{i,d}(t) \leq D_2(t)$, learner i asks j_i to send it $N_C^{j_i,d}(t)$

Initialize($\mathcal{A}$):

1: **for** $C \in \mathcal{A}$ **do**
2: 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 $k \in \mathcal{M}_{-i}$.
3: **end for**

Figure 4: Pseudocode of the initialization module.

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 to learner j_i. If C is not activated by j_i yet, then it sends $N_C^{j_i,d}(t) = 0$ and activates the hypercube C for its type-d context. Then learner i sets $N_{1,j_i,C}^{i,d}(t) = N_C^{j_i,d}(t) - N_{2,j_i,C}^{i,d}(t)$ and checks again if $N_{1,j_i,C}^{i,d}(t) \leq D_2(t)$. 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 classification function in $\mathcal{F}_j$ it had chosen to make a prediction for learner i. If $N_{1,j_i,C}^{i,d}(t) > D_2(t)$, 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 $N_{1,j_i,C}^{i,d}(t) > D_2(t)$ 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 $N_{1,j_i,C}^{i,d}(t) > D_2(t)$ and $N_{2,j_i,C}^{i,d}(t) \leq D_3(t)$, and updates $N_{2,j_i,C}^{i,d}(t)$ and the sample mean accuracy of learner j_i which is ratio of the total number of correct predictions j_i has made for i's contexts in hypercube C to the total number of predictions j_i has made for i for contexts in hypercube C. For simplicity of notation we let $N_{j_i,C}^{i,d}(t) := N_{2,j_i,C}^{i,d}(t)$ for $j_i \in \mathcal{M}_{-i}$. Let $\mathcal{S}_{C_i^d(t)}^{i,d}(t) := \{k_i \in \mathcal{F}_i$ **such that** $N_{k_i,C_i^d(t)}^{i,d}(t) \leq D_1(t)$ **or** $j_i \in \mathcal{M}_{-i}$ **such that** $N_{1,j_i,C_i^d(t)}^{i,d}(t) \leq D_2(t)$ **or** $N_{2,j_i,C_i^d(t)}^{i,d}(t) \leq D_3(t)\Big\}$. and $\mathcal{S}_{\boldsymbol{C}_i(t)}^{i}(t) = \cup_{d\in\mathcal{D}} \mathcal{S}_{C_i^d(t)}^{i,d}(t)$. If $S_{\boldsymbol{C}_i(t)}^{i}(t) \neq \emptyset$ then ACAP randomly selects an arm in $S_{\boldsymbol{C}_i(t)}^{i}(t)$ to explore, while if $S_{\boldsymbol{C}_i(t)}^{i}(t) = \emptyset$, ACAP selects an arm in $\arg\max_{k\in\mathcal{K}_i} \left(\max_{d\in\mathcal{D}} \bar{r}_{k,C_i^d(t)}^{i,d}(t)\right)$, where $\bar{r}_{k,C_i^d(t)}^{i,d}(t)$ is the sample mean of the rewards collected from arm k in time steps for which the type-d context is in $C_i^d(t)$ from the activation of $C_i^d(t)$ 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 steps for which the type-d context is in $C_i^d(t)$ from the activation of $C_i^d(t)$ to time t for $k \in \mathcal{M}_{-i}$.

Train(t, k, $\boldsymbol{N}_{1,k}^i$):

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

Explore(t, k, $\boldsymbol{N}_k^i$, $\bar{\boldsymbol{r}}_k^i$):

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

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

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

Figure 5: Pseudocode of the training, exploration and exploitation modules.

4.2 Analysis of the regret of ACAP In this subsection we analyze the regret of ACAP for arbitrary context arrival processes and derive a sublinear upper bound on the regret. 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 4.1. *All the active hypercubes $\mathcal{A}_i^d(t)$ 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 4.2. *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 step.

Lemma 4.2 states that the regret due to trainings and explorations increases exponentially with z.

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 $\underline{\mu}_{k,\boldsymbol{C}} := \min_{d \in \mathcal{D}} \underline{\mu}^d_{k,C^d}$, 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 $\boldsymbol{C}$. In the next lemma we bound the regret due to choosing a suboptimal arm in the exploitation steps.

LEMMA 4.3. *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 steps $1 \leq t \leq T$ in exploitation steps 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 $k_i \in \mathcal{F}_i$, let $\mathcal{E}^i_{k_i,\boldsymbol{C}}(t)$ be the set of rewards collected by learner i from arm k_i in time steps when the context vector of learner i is in the active set $\boldsymbol{C}$ by time t. For $j_i \in \mathcal{M}_{-i}$, let $\mathcal{E}^i_{j_i,\boldsymbol{C}}(t)$ be the set of rewards collected from selections of learner j_i in time steps $t' \in \{1, \ldots, t\}$ 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 $k_i \in \mathcal{F}_i$ let $\mathcal{B}^i_{k_i,\boldsymbol{C}}(t) := \Omega$. In order to facilitate our analysis of the regret, we generate two different artificial i.i.d. processes to bound the probabilities related to $\bar{r}^{i,d}_{k,C^d}(t)$, $k \in \mathcal{K}_i$. 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 $r^{b,i,d}_{k,C^d}(t)$ denote the sample mean of the t samples from the best process and $r^{w,i,d}_{k,C^d}(t)$ denote the sample mean of the t samples from the worst process. We have for any $k \in \mathcal{L}^i_{\boldsymbol{C},B}$

$$
\begin{aligned}
&P\left(\mathcal{V}^i_{k,\boldsymbol{C}}(t), \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \\
&\leq P\left(\max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) \geq \overline{\mu}_{k,\boldsymbol{C}} + H_t, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) \\
&+ P\Big(\max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) \geq \bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t)) \\
&-2t^{\phi-1}, \max_{d \in \mathcal{D}} \bar{r}^{b,i,d}_{k,C^d}(N^{i,d}_{k,C^d}(t)) < \overline{\mu}_{k,\boldsymbol{C}} + L2^{-l_{\max}(\boldsymbol{C})\alpha} \\
&+H_t + 2t^{\phi-1}, \bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t)) \\
&> \underline{\mu}_{k^*(\boldsymbol{C}),\boldsymbol{C}} - L2^{-l_{\max}(\boldsymbol{C})\alpha} - H_t, \mathcal{W}^i_{\boldsymbol{C}}(t)\Big) \qquad (4.1) \\
&+ P\left(\bar{r}^{w,i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(N^{i,d^*(\boldsymbol{C})}_{k^*(\boldsymbol{C}),C^{d^*(\boldsymbol{C})}}(t)) \leq \underline{\mu}_{k^*(\boldsymbol{C}),\boldsymbol{C}} - H_t\right. \\
&\left.+2t^{\phi-1}, \mathcal{W}^i_{\boldsymbol{C}}(t)\right) + P((\mathcal{B}^i_{k,\boldsymbol{C}}(t))^c),
\end{aligned}
$$

where $H_t > 0$. In order to make the probability in (4.1) equal to 0, we need

$$4t^{\phi-1} + 2H_t \leq (B-2)L2^{-l_{\max}(\boldsymbol{C})\alpha}. \qquad (4.2)$$

By Lemma 4.1, (4.2) holds when

$$4t^{\phi-1} + 2H_t \leq (B-2)L2^{-\alpha}t^{-\alpha/p}. \qquad (4.3)$$

For $H_t = 4t^{\phi-1}$, $\phi = 1 - z/2$, $z \geq 2\alpha/p$ and $B = 12/(L2^{-\alpha}) + 2$, (4.3) holds by which (4.1) 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

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

We also have $P(\mathcal{B}^i_{k_i,\boldsymbol{C}}(t)^c) = 0$ for $k_i \in \mathcal{F}_i$ 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 steps 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 $k_i \in \mathcal{F}_i$ 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 classification functions when called by learner i in exploitation steps when the context vector of learner i belongs to is $\boldsymbol{C}$.

LEMMA 4.4. *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 steps $t' \in \{1,\ldots,T\}$, the regret due to a near optimal learner choosing a suboptimal classification function 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 4.5. *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 classification functions 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 classification functions 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 $k_i \in \mathcal{F}_i$ 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 4.6. *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 steps $1 \leq t \leq T$ 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

$$A\sum_{l=1}^{l_{\max,u}-1} 2^l 2^{pl} < T \tag{4.4}$$

otherwise the highest level hypercube by time T will be $l_{\max,u}-1$. Solving (4.4) 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 4.6, 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 classification function for similar contexts. Next, we combine the results from Lemmas 4.2, 4.3 and 4.6 to obtain the regret bound for ACAP.

THEOREM 4.1. *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 training and explorations is bounded by Lemma 4.2. 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 4.6 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. 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 4.3.

From the result of Theorem 4.1, it is seen that the regret increases linearly with the number of learners in the system and their number of classification functions (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 classification function, 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 classification functions 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 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 [5] for algorithms which do not adaptively choose the type of the context to exploit, whose time order approaches linear as D increases.

5 Numerical Results

For our simulations, we use the network security data from KDD Cup 1999 data set. We compare the performance of our learning algorithms with state-of-the-art online ensemble learning techniques given in Table 3. Different from our algorithm ACAP which makes a prediction based on a single classification function at each time step, these algorithms combine the predictions of all classification functions of all learners to make the final prediction. The network security data has 42 features. The goal is to predict at any given time if an attack occurs or not based on the values of the features. To show the effectiveness of adaptively choosing the best context over time we take $D = 3$; type-1 context is the label at the previous time step, type-2 context is the feature named *srcbytes*, which is the number of data bytes from source to destination, type-3 context is time. All the type-d context information is normalized to be in $[0, 1]$. There are 4 local learners. Each local learner has 2 classification functions. The classification costs c^i_k are set to 0 for all $k \in \mathcal{K}_i$. All classification functions are trained using 5000 consecutive samples from different segments of the network security data (except for OnAda in which the base classifiers are trained online). Then, they are tested on $T = 20000$ consecutive samples. In our first simulation S1, there are two good classifiers that have low number of errors on the test data, while in our second simulation S2, there are no good classifiers. The types of classification functions used in S1 and S2 are given in Table 2 along with the number of errors each of these classification functions made on the test data. From Table 2, we can observe that the error percentage of the best classification function is 3 in S1, while it is 47 in S2. A situation like in S2 can appear when the distribution of the data changes abruptly, i.e., concept drift, so that the classification functions trained on the old data becomes inaccurate for the new data. In our numerical results, we will show how the context information can be used to improve the performance in both S1 and S2. The accuracies of the classifiers on the test data are unknown to the learners. In all our simulations, we assume that the test data sequentially arrives to the system and the label is revealed to the algorithms

Learner	Class. Fn. (S1)	Error % (S1)	Class. Fn. (S2)	Error % (S2)
1	Naive Bayes	47	Naive Bayes	47
	Logistic	3	Random	50
2	Always 1	53	Always 1	53
	Voted Perceptron	4	Random	50
3	RBF Network	47	RBF Network	47
	J48	47	J48	47
4	Random Tree	47	Random Tree	47
	Always 0	47	Always 0	47

Table 2: Base classification functions used by the learners and their error percentages on the test data for S1 and S2.

with a one step delay.

Results in Table 3 show that ACAP has a very small error percentage for both S1 and S2 and is significantly better than all ensemble learning techniques except TrackExp for S2. Recall that ACAP's prediction at any t is only based on a single classification function. Results for ACAP-d shows that performance is significantly affected by what is chosen as the context if contexts are not adaptively chosen over time. Clearly for our specific example, previous label is the best context to exploit. However, in general the best type of context can change over time since the data stream can be non-stationary, and it is impossible to predict a priori which type of context is the best to exploit. ACAP is slightly worse than ACAP-1 due to the fact that it needs to train and explore for all three types of context, while ACAP-1 does this only for one type of context. Although we do not check if the conditions of Theorem 4.1 is satisfied for the set of parameters used by ACAP, our experimentation with different types of parameters (omitted due to space) suggests that the training and exploration rates can be made much smaller than the ones stated in Theorem 4.1 without degrading the performance.

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

[6] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 5, pp. 730–742, 2010.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

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

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

[21] 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.

[22] A. Slivkins, "Contextual bandits with similarity infor-

Abbrev.	Name	Ref.	Parameters	Error%(S1)	Error%(S2)
BC	Best classification function in $\mathcal{F}$	-	-	3.12	46.9
Ada	Adaboost	[24]	-	4.82	53.1
OnAda	Fan's Online Adaboost	[25]	Window size $w = 100$	2.69	2.42
AM	Average Majority	[1]	-	46.9	46.9
Blum	Blum's variant of weighted majority	[26]	Multiplicative parameters $\beta = 0.5$, $\gamma = 1.5$	53.1	2.82
TrackExp	Herbster's variant of weighed majority	[27]	Multiplicative and sharing parameters $\beta = 0.5$, $\alpha = 0.25$	2.41	0.64
ACAP	Adaptive contexts and adaptive partitions algorithm $D = 3$	Our work	$D_1(t) = D_3(t) = 1/8t^{1/8} \log t$ $D_2(t) = 1/4t^{1/8} \log t$, $A = 1$, $p = 4$	0.7	0.83
ACAP-1	**ACAP** $D = 1$, context = previous label	"	Same as **ACAP**	0.39	0.45
ACAP-2	**ACAP** $D = 1$, context = *srcbytes* feature	"	Same as **ACAP**	3.75	41.3
ACAP-2	**ACAP** $D = 1$, context = time	"	Same as **ACAP**	20.5	11.7
Bandit-T	Similar to **ACAP** with training phase but does not exploit contexts	"	Same as **ACAP**	5.2	49.8

Table 3: Error percentages of various online learning methods and our work for settings S1 and S2.

mation," in *24th Annual Conference on Learning Theory (COLT)*, 2011.

[23] 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.

[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*. Springer, 1995, pp. 23–37.

[25] W. Fan, S. J. Stolfo, and J. Zhang, "The application of adaboost for distributed, scalable and on-line learning," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 362–366.

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

[27] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.