**Fifty-first Annual Allerton Conference**
**Allerton House, UIUC, Illinois, USA**
**October 2 - 3, 2013**

# Distributed Online Big Data Classification Using Context Information

Cem Tekin, Mihaela van der Schaar

*Abstract*— **Distributed, online data mining systems have emerged as a result of applications requiring analysis of large amounts of correlated and high-dimensional data produced by multiple distributed data sources. We propose a distributed online data classification framework where data is gathered by distributed data sources and processed by a heterogeneous set of distributed learners which learn online, at run-time, how to classify the different data streams either by using their locally available classification functions or by helping each other by classifying each other's data. Importantly, since the data is gathered at different locations, sending the data to another learner to process incurs additional costs such as delays, and hence this will be only beneficial if the benefits obtained from a better classification will exceed the costs. We model the problem of joint classification by the distributed and heterogeneous learners from multiple data sources as a distributed contextual bandit problem where each data is characterized by a specific context. We develop a distributed online learning algorithm for which we can prove sublinear regret. Compared to prior work in distributed online data mining, our work is the first to provide analytic regret results characterizing the performance of the proposed algorithm.**

## I. Introduction

A plethora of Big Data applications (network security, surveillance, health monitoring etc.) are emerging which require online classification of large data sets collected from distributed network and traffic monitors, multimedia sources, sensor networks, etc. This data is heterogeneous and dynamically evolves over time. In this paper, we introduce a distributed online learning framework for classification of high-dimensional data collected by distributed data sources.

The distributedly collected data is processed by a set of decentralized heterogeneous learners equipped with classification functions with unknown accuracies. In this setting communication, computation and sharing costs make it infeasible to use centralized data mining techniques where a single learner can access the entire data set. For example, in a wireless sensor surveillance network, nodes in different locations collect different information about different events. The learners/machines at each node of the network may run different classification algorithms, may have different resolution, processing speed, etc.

The input data stream and its associated context can be time-varying and heterogeneous. We use the term "context" generically, to represent any information related to the input data stream such as time, location and type (e.g., data features/characteristics/modality) information. 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 our proposed approach is the focus on how the context information of the captured data can be 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 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 a distributed online learning algorithm whose long-term average reward converges 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 define the regret as the difference between the expected total reward of the best distributed classification scheme given complete knowledge about classification function accuracies and the expected total reward of the algorithm used by each learner. We prove a sublinear upper bound on the regret, which implies that the average reward converges to the optimal average reward. The upper bound on regret gives a lower bound on convergence rate to the optimal average reward. To illustrate our approach, we provide numerical results by applying our learning algorithm to the classification of network security data and compare the results with existing state-of-the-art solutions.

The remainder of the paper is organized as follows. In Section II we describe the related work and highlight the differences from our work. In Section III we describe the decentralized data classification problem, the optimal distributed classification scheme given the complete system

C. Tekin and M. van der Schaar are with the Dept. of Electrical Engineering, University of California, Los Angeles, CA, cmtkn@ucla.edu, mihaela@ee.ucla.edu

model, its computational complexity, and the regret of a learning algorithm with respect to the optimal classification scheme. Then, we consider the model with unknown system statistics and propose a distributed online learning algorithm with *uniform contextual partitioning* in Section IV. Using a network security application we provide numerical results on the performance of our distributed online learning algorithm in Section V. Finally, the concluding remarks are given in Section VI.

## II. Related Work

Related work can be divided into two categories: Online learning for data mining and multi-armed bandit methods aimed at learning how to act sequentially.

Online learning in distributed data classification systems aims to address the informational decentralization, communication costs and privacy issues arising in these systems. For example in [1]–[4], various solutions are proposed for distributed data mining problems of horizontally distributed data, while in [5], [6] 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 [6]–[9], 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.

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 [3], [4], [10]–[15]) propose ensemble learning techniques including bagging, boosting, stacked generalization and cascading, where the goal is to use classification results from several classifiers to increase the prediction accuracy. 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 method can easily be adapted to perform ensemble learning. We provide a detailed comparison to our work in Table I.

Other than distributed data mining, our learning framework can be applied to any problem that can be formulated as a decentralized contextual bandit problem. Contextual bandits have been studied before in [16]–[19] in a single agent setting, where the agent sequentially chooses from a set of alternatives with unknown rewards, and the rewards depend on the context information provided to the agent at each time step. To the best of our knowledge, our work is the first to address the decentralized contextual bandit problem in a system of cooperative learning agents. In [20], a contextual bandit algorithm named LinUCB is proposed for recommending personalized news articles, which is variant of the UCB algorithm [21] designed for linear payoffs. Numerical results on real-world Internet data are provided, but no theoretical results on the resulting regret are derived. A perceptron based algorithm is used with upper confidence bounds in [22] in a centralized single user setting that achieves sublinear regret when the instances are chosen by an adversary and the learning algorithm receives binary feedback about the true label instead of the true label itself. Previously, distributed multi-user learning is only considered in the standard finite armed bandit problem. In [23], [24] distributed online learning algorithms that converge to the optimal allocation with logarithmic regret are proposed, given that the optimal allocation is an orthogonal allocation in which each user selects a different action. This is generalized in [25] to dynamic resource sharing problems and logarithmic regret results are also proved for this case. Alternatively, in this paper, we consider distributed online learning in a contextual bandit setting. We provide a detailed comparison between our work and related work in multi-armed bandit learning in Table II. Our decentralized contextual learning framework can be seen as an important extension of the centralized contextual bandits framework [16]. The main difference is that: (i) a three phase learning algorithm with *training*, *exploration* and *exploitation* phases are needed instead of the standard two phase, i.e., *exploration* and *exploitation* phases, algorithms used in centralized contextual bandit problems; (ii) the adaptive partitions of the context space should be formed in a way that each learner can efficiently utilize what is learned by other learners about the same context. In the distributed contextual framework, the training phase is necessary since the context arrivals to learners are different which makes the learning rates of the learners for different context different.

## III. 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 $\mathcal{M}_{-i} = \mathcal{M} - \{i\}$. 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 context $x_i(t)$ arrives to each learner $i \in \mathcal{M}$, (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 its sent its data and context, (iii) the truth (true label) is revealed eventually, perhaps by

|  | [3], [13] [8], [14], [15] | [7], [9] | [5] | 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 |
| Regret bound | no | no | no | yes - sublinear |

TABLE I

COMPARISON WITH RELATED WORK IN DISTRIBUTED DATA MINING

|  | [16]–[19] | [23]–[25] | [26] | This work |
|---|---|---|---|---|
| Multi-user | no | yes | yes | yes |
| Cooperative | N/A | yes | no | yes |
| Contextual | yes | no | no | yes |
| Data arrival process | arbitrary | i.i.d. or Markovian | i.i.d. | arbitrary |
| Regret | sublinear | logarithmic | may be linear | sublinear |

TABLE II

COMPARISON WITH RELATED WORK IN MULTI-ARMED BANDITS

events or by a supervisor, only to the classifier where the data arrived.

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. Classifier $i$ knows the functions in $\mathcal{F}_i$ and costs of calling them[1], but not their accuracies, while it knows the set of other learners $\mathcal{M}_{-i}$ and costs of calling them but does not know the functions $\mathcal{F}_k$, $k \in \mathcal{M}_{-i}$, but only knows an upper bound on the number of classification functions that each learner has, i.e., $F_{\max}$ on $|\mathcal{F}_k|$[2], $k \in \mathcal{M}_{-i}$. Classifier $i$ 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 $k \in \mathcal{F}_i$ incurs a cost $d_k$. 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. Since the costs are bounded, without loss of generality we assume that costs are normalized, i.e., $d_k \in [0, 1]$. 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 input 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 $k \in \mathcal{M}_{-i}$ a cost of $d_k$ is incurred. The learners are cooperative which implies that learner $k \in \mathcal{M}_{-i}$ will return a label to $i$ when called by

[1]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.

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

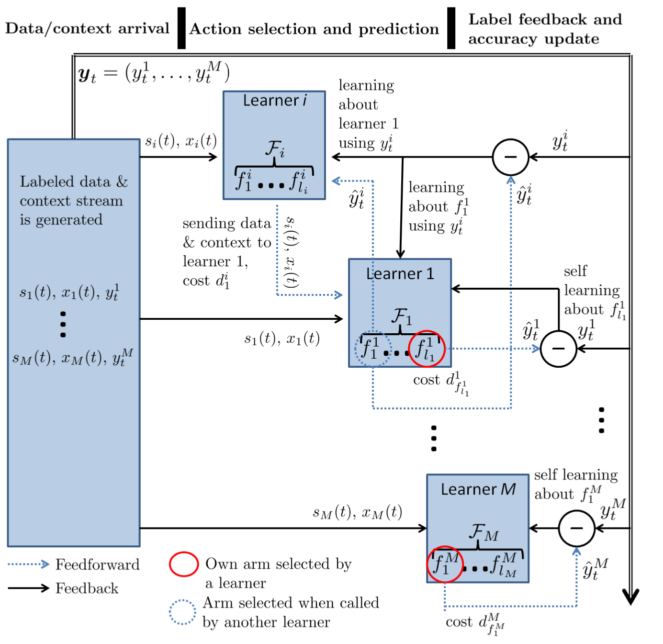
Data/context arrival
Action selection and prediction
Label feedback and accuracy update
$\boldsymbol{y}_t = (y_t^1, \ldots, y_t^M)$
Learner $i$
$\mathcal{F}_i$
$f_1^i \ldots f_{l_i}^i$
learning about learner 1 using $y_t^i$
$y_t^i$
$\hat{y}_t^i$
$s_i(t), x_i(t)$
Labeled data & context stream is generated
$s_1(t), x_1(t), y_t^1$
$s_M(t), x_M(t), y_t^M$
sending data & context to learner 1, cost $d_1^i$
$s_i(t), x_i(t)$
learning about $f_1^1$ using $y_t^i$
Learner 1
$\mathcal{F}_1$
$f_1^1 \ldots f_{l_1}^1$
self learning about $f_{l_1}^1$
$y_t^1$
$\hat{y}_t^1$
$s_1(t), x_1(t)$
cost $d_{f_{l_1}^1}^1$
Learner $M$
$\mathcal{F}_M$
$f_1^M \ldots f_{l_M}^M$
self learning about $f_1^M$
$y_t^M$
$\hat{y}_t^M$
$s_M(t), x_M(t)$
cost $d_{f_1^M}^M$
Feedforward
Feedback
Own arm selected by a learner
Arm selected when called by another learner

Fig. 1. Operation of the distributed data classification system from the viewpoint of learner 1.

$i$. Similarly, when called by $k \in \mathcal{M}_{-i}$, learner $i$ will return a label to $k$. 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 $d_k$ also captures the cost to learner $k$ to classify and sent the label to learner $i$, then maximizing $i$'s own utility corresponds to maximizing the system utility. Let $\mathcal{K}_i = \mathcal{F}_i \cup \mathcal{M}_{-i}$. We call $\mathcal{K}_i$ the set of arms (alternatives).

We assume that each classification function produces a binary label[3]. The data stream at time $t$ arrives to learner $i$ with context information $x_i(t)$. The context may be generated as a result of pre-classification or a header of the data stream. For simplicity we assume that the context space is $\mathcal{X} = [0, 1]^d$, while our results will hold for any bounded $d$ dimensional context space. We also note that the data input is high dimensional and its dimension is greater than $d$ (in most of the cases its much larger than $d$) . For example, the network security data we use in numerical results section has 42 features, while the dimension of the context we use is at most 1.

Each classification function $k \in \cup_{i \in \mathcal{M}} \mathcal{F}_i$ has an unknown accuracy $\pi_k(x) \in [0, 1]$, depending on the context $x$. The accuracy $\pi_k(x)$ represents the probability that an input stream with context $x$ will be labeled correctly when classification function $k$ is used to label it. Different classification functions can have different accuracies for the same context. Although we do not make any assumptions about the classification accuracy $\pi_k(x)$ and the classification cost $d_k$, in general one can assume that classification accuracy increases with

[3]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.

classification cost (e.g., classification functions with higher resolution, better processing). We assume that each classifier has similar accuracies for similar contexts; we formalize this in terms of a (uniform) Lipschitz condition.

*Assumption 1:* For each $k \in \cup_{i \in \mathcal{M}} \mathcal{F}_i$, there exists $L > 0$, $\alpha > 0$ such that for all $x, x' \in \mathcal{X}$, we have $|\pi_k(x) - \pi_k(x')| \leq L||x - x'||^\alpha$, where $||.||$ denotes the Euclidian norm in $\mathbb{R}^d$.

Assumption 1 indicates that the accuracy of a classification function for similar contexts will be similar to each other. We assume that $\alpha$ is known by the learners. In the bandit setting this is referred to as *similarity information* [16], [27].

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 $x_t$ 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 [16]–[19]. 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 subsection we formally define the benchmark solution which is computed using perfect knowledge about classification accuracies. Then, we define the regret which is the performance loss due to uncertainty about classification accuracies.

### A. Optimal Classification with Complete Information

Our benchmark when evaluating the performance of the learning algorithm is the optimal solution which selects the classification function $k$ with the highest accuracy minus cost for learner $i$ from the set $\cup_{j \in \mathcal{M}} \mathcal{F}_j$ given context $x_t$ 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 $k^*(x) = \arg\max_{k \in \mathcal{K}_i} \pi_k(x) - d_k, \quad \forall x \in \mathcal{X}$.

Knowing the optimal solution means that learner $i$ knows the classification function in $\cup_{i \in \mathcal{M}} \mathcal{F}_i$ that yields the highest expected accuracy for each $x \in \mathcal{X}$. Choosing the best classification function for each context $x$ requires to evaluate the accuracy minus cost for each context and is computationally intractable, because the context space $\mathcal{X}$ has infinitely many elements.

### B. The Regret of Learning

In this subsection we define the regret as a performance measure of the learning algorithm used by the learners. Simply, the regret is the loss incurred due to the unknown system dynamics. Regret of a learning algorithm for learner $i$ is defined with respect to the best arm $k^*(x)$ for learner $i$. The regret of a learning algorithm is given by

$$R(T) := \sum_{t=1}^{T} \pi_{k^*(x_t)}(x_t) - E\left[\sum_{t=1}^{T} (I(\hat{y}_t^i = y_t) - d_{k(x_t)})\right],$$

where $\hat{y}_t^i$ denotes prediction made by the classification function or other learner $k(x_t)$ chosen by learner $i$ at time $t$, $y_t$ denotes the true label and the expectation is taken with respect to the random selection made by the learning algorithm. In the following section we will propose a distributed learning algorithm with sublinear regret.

## IV. A Distributed Uniform Context Partitioning Algorithm

In this section we consider a uniform partitioning algorithm. Assume that each learner runs the learning algorithm *Classify or Send for Classification* (CoS) given in Fig. 2. Let $m_T$ be the *slicing parameter* of CoS that determines the partition of the context space $\mathcal{X}$. Basically, choosing a large $m_T$ will improve classification accuracy while increasing the number of explorations. We will analyze the performance for a fixed $m_T$ and then optimize over it. CoS forms a partition of $[0,1]^d$ consisting of $(m_T)^d$ sets where each set is a $d$-dimensional hypercube with dimensions $1/m_T \times 1/m_T \times \ldots \times 1/m_T$. Let $\mathcal{P}_T = \{P_1, P_2, \ldots, P_{(m_T)^d}\}$ denote this partition where each $P_l$ is a hypercube. When clear from the context, we will use $l$ instead of $P_l$ to denote the hypercube. The set of arms for learner $i$ consists of its classification functions and the set of learners it can send the data to, which is denoted by $\mathcal{K}_i$.

```
Classify or Send for Classification (CoS for learner $i$):

1: Input: $D_1(t)$, $D_2(t)$, $D_3(t)$, $T$, $m_T$
2: Initialize: Partition $[0,1]^d$ into $(m_T)^d$ sets. Let
   $\mathcal{P}_T = \{P_1, \ldots, P_{(m_T)^d}\}$ denote the sets in this partition.
   $N_{k,l}^i = 0, \forall k \in \mathcal{K}_i, P_l \in \mathcal{P}_T$,
   $N_{1,k,l}^i = 0, \forall k \in \mathcal{M}_{-i}, P_l \in \mathcal{P}_T$.
3: while $t \geq 1$ do
4:   for $l = 1, \ldots, (m_T)^d$ do
5:     if $x_i(t) \in P_l$ then
6:       if $\exists k \in \mathcal{F}_i$ such that $N_{k,l}^i \leq D_1(t)$ then
7:         Run Explore($k$, $N_{k,l}^i$, $\bar{r}_{k,l}$)
8:       else if $\exists k \in \mathcal{M}_{-i}$ such that $N_{1,k,l}^i \leq D_2(t)$ then
9:         Obtain $N_l^k(t)$ from $k$, set
           $N_{1,k,l}^i = N_l^k(t) - N_{k,l}^i$
10:        if $N_{1,k,l}^i \leq D_2(t)$ then
11:          Run Train($k$, $N_{1,k,l}^i$)
12:        else
13:          Go to line 15
14:        end if
15:      else if $\exists k \in \mathcal{M}_{-i}$ such that $N_{k,l}^i \leq D_3(t)$ then
16:        Run Explore($k$, $N_{k,l}^i$, $\bar{r}_{k,l}$)
17:      else
18:        Run Exploit($\boldsymbol{M}_l^i$, $\bar{\boldsymbol{r}}_l$, $\mathcal{K}_i$)
19:      end if
20:    end if
21:  end for
22:  $t = t + 1$
23: end while
```

Fig. 2. Pseudocode of CoS algorithm.

For each set in the partition $\mathcal{P}_T$, learner $i$ keeps several counters for each arm in $\mathcal{K}_i$. Any time step $t$ can be in one of the three phases: *training* phase in which learner $i$ trains another learner by sending its own data, *exploration* phase in which learner $i$ updates the estimated reward of an arm in $\mathcal{K}_i$ by selecting it, and *exploitation* phase in which learner $i$ selects the arm with the highest estimated reward. The pseudocodes of these phases are given in Fig. 3. Upon each data arrival, learner $i$ first checks to which set in the partition $\mathcal{P}_T$ the context belongs. Let $N_l^i(t)$ be the number of data arrivals in $P_l$ of learner $i$ by time $t$. For $k \in \mathcal{F}_i$, let $N_{k,l}^i(t)$ be the number of times arm $k$ is selected in response to a data

```
**Train**($k$, $n$):
  1: select arm $k$
  2: Receive reward $r_k(t) = I(\hat{y}_t^i = y_t) - d_{k(x_i(t))}$
  3: $n++$
**Explore**($k$, $n$, $r$):
  1: select arm $k$
  2: Receive reward $r_k(t) = I(\hat{y}_t^i = y_t) - d_{k(x_i(t))}$
  3: $r = \frac{nr + r_k(t)}{n+1}$
  4: $n++$
**Exploit**($\boldsymbol{n}$, $\boldsymbol{r}$, $\mathcal{K}_i$):
  1: select arm $k \in \arg\max_{j \in \mathcal{K}_i} r_j$
  2: Receive reward $r_k(t) = I(\hat{y}_t^i = y_t) - d_{k(x_i(t))}$
  3: $\bar{r}_k = \frac{n_k \bar{r}_k + r_k(t)}{n_k + 1}$
  4: $n_k++$
```

Fig. 3. Pseudocode of the training, exploration and exploitation modules.

arriving to set $P_l$ in the partition $\mathcal{P}_T$ by learner $i$ by time $t$. Note that learner $i$ does not know anything about learner $k$'s classification functions. Therefore, before forming estimates about the reward of $k$, it needs to make sure that $k$ will almost always select its optimal classification function when called by $i$. This is why the training phase is needed for learners $k \in \mathcal{M}_{-i}$. To separate training, exploration and exploitation phases, learner $i$ keeps two counters for $k \in \mathcal{M}_{-i}$. The first one, i.e., $N^i_{1,k,l}(t)$, counts the number of data arrivals to learner $k$ in set $l$ by time $t$ which includes data arrivals with context $x_k(t') \in P_l$, $t' < t$ and data arrivals from learner $i$ to $k$ in the training phases of $i$. The second one, i.e., $N^i_{2,k,l}(t)$, counts the number of data arrivals to learner $k$ that are used in $i$'s reward estimation of $k$. This is the number of times data is sent from learner $i$ to learner $k$ in the exploration phase or exploitation phase of learner $i$. For simplicity of notation we let $N^i_{k,l}(t) := N^i_{2,k,l}(t)$ for $k \in \mathcal{M}_{-i}$. The values of these counters are random variables when the context arrival process is stochastic. Based on the values of these counters at time $t$, learner $i$ either trains, explores or exploits an arm in $\mathcal{K}_i$. This three phase learning structure is one of the major components of our learning algorithm which makes it different than the algorithms proposed for the contextual bandits in the literature which only have exploration and exploitation phases.

When a context $x_i(t) \in P_l$ arrives, in order to make sure that all classification functions of all learners are explored sufficiently, learner $i$ checks if the following set is nonempty.

$$\mathcal{S}_{i,l} := \big\{k \in \mathcal{F}_i \text{ such that } N^i_{k,l}(t) \le D_1(t) \text{ or } k \in \mathcal{M}_{-i} \\ \text{such that } N^i_{1,k,l}(t) \le D_2(t) \text{ or } N^i_{2,k,l}(t) \le D_3(t)\big\}.$$

For $k \in \mathcal{M}_{-i}$, let $\mathcal{E}^i_{k,l}(t)$ be the set of rewards collected from selections of arm $k$ in set $l$ by time $t$ for which $N^i_{1,k,l}(t) > D_2(t)$. We note that, learner $i$ does not have to communicate with learner $k$ at each time step to update $N^i_{1,k,l}(t)$. It only needs to communicate when $N^i_{1,k,l}(t) \le D_2(t)$. To obtain the correct value of $N^i_{1,k,l}(t)$ it needs to know $N^k_l(t)$, since $N^i_{1,k,l}(t) = N^k_l(t) - N^i_{2,k,l}(t)$. For $k \in \mathcal{F}_i$, let $\mathcal{E}^i_{k,l}(t)$ the set of rewards collected from arm $k$ by time $t$. If $\mathcal{S}_{i,l} \neq \emptyset$, then learner $i$ explores by choosing randomly an arm $\alpha(t) \in \mathcal{S}_{i,l}$. If $\mathcal{S}_{i,l} = \emptyset$, this implies that all classification functions have been explored sufficiently, so that learner $i$ exploits by choosing the arm with the highest sample mean estimate, i.e.,

$$\alpha(t) \in \arg\max_{k \in \mathcal{K}_i} \bar{r}^i_{k,l}(t), \quad (1)$$

where $\bar{r}^i_{k,l}(t)$ is the sample mean of the rewards in set $\mathcal{E}^i_{k,l}(t)$. Explicitly, $\bar{r}^i_{k,l}(t) = (\sum_{r \in \mathcal{E}^i_{k,l}(t)} r)/|\mathcal{E}^i_{k,l}(t)|$, where each $r \in \mathcal{E}^i_{k,l}(t)$ is equal to $1 - d_k$ if the classification is correct and $-d_k$ if the classification is wrong. When there is more than one maximizer of (1), one of them is randomly selected. The exploration control functions $D_1(t)$, $D_2(t)$ and $D_3(t)$ ensure that each classification function is selected sufficiently many number of times so that the sample mean estimates $\bar{r}^i_{k,l}(t)$ are accurate enough. In the following subsection we prove an upper bound on the regret of CoS.

## *A. Analysis of the regret of CoS*

Let $\mu_k(x) = \pi_k(x) - d_k$, and $\beta_a = \sum_{t=1}^{\infty} 1/t^a$. For each $P_l \in \mathcal{P}_T$ let $\overline{\mu}_{k,l} := \sup_{x \in P_l} \mu_k(x)$ and $\underline{\mu}_{k,l} := \inf_{x \in P_l} \mu_k(x)$. Let $x^*_l$ be the context at the center of the hypercube $P_l$. We define the optimal arm for $P_l$ as $k^*(l) := \arg\max_{k \in \mathcal{K}_i} \mu_k(x^*_l)$. Let $\mathcal{L}^i_\theta(t) := \left\{k \in \mathcal{K}_i \text{ such that } \underline{\mu}_{k^*(l),l} - \overline{\mu}_{k,l} > a_1 t^\theta\right\}$, be the set of suboptimal arms for learner $i$ at time $t$, where $\theta < 0$, $a_1 > 0$. The learners are not required to know the values of the parameters $\theta$ and $a_1$. They are only used in our analysis of the regret. First, we will give regret bounds that depend on values of $\theta$ and $a_1$ and then we will optimize over these values to find the best bound.

The regret can be written as a sum of three components: $R(T) = E[R_e(T)] + E[R_s(T)] + E[R_n(T)]$, where $R_e(T)$ is the regret due to training and explorations by time $T$, $R_s(T)$ is the regret due to suboptimal arm selections in exploitations by time $T$ and $R_n(T)$ is the regret due to near optimal arm selections in exploitations by time $T$, which are all random variables. In the following lemmas we will bound each of these terms separately. The following lemma bounds $E[R_e(T)]$. Due to space constraints some of the proofs are not included in the paper. For the complete proofs please see the online appendix [28].

*Lemma 1:* When CoS is run by learner $i$ with parameters $D_1(t) = t^z \log t$, $D_2(t) = F_{\max} t^z \log t$, $D_3(t) = t^z \log t$ and $m_T = \lceil T^\gamma \rceil$[4], where $0 < z < 1$ and $0 < \gamma < 1/d$, we have

$$\begin{aligned} E[R_e(T)] &\le \sum_{l=1}^{(m_T)^d} (|\mathcal{F}_i| + (M-1)(F_{\max}+1)) T^z \log T \\ &\quad + (|\mathcal{F}_i| + 2(M-1))(m_T)^d \\ &\le 2^d (|\mathcal{F}_i| + (M-1)(F_{\max}+1)) T^{z+\gamma d} \log T \\ &\quad + 2^d (|\mathcal{F}_i| + 2(M-1)) T^{\gamma d}. \end{aligned}$$

From Lemma 1, we see that the regret due to explorations is linear in the number of hypercubes $(m_T)^d$, hence exponential in parameter $\gamma$ and $z$. We conclude that $z$ and $\gamma$ should be small enough to achieve sublinear regret in exploration steps. For any $k \in \mathcal{K}_i$ and $P_l \in \mathcal{P}_T$, the sample mean $\bar{r}_{k,l}(t)$

[4]For a number $r \in \mathbb{R}$, let $\lceil r \rceil$ be the smallest integer that is greater than or equal to $r$.

represents a random variable which is the average of the independent samples in set $\mathcal{E}^i_{k,l}(t)$. Different from classical finite-armed bandit theory [21], these samples are not identically distributed. In order to facilitate our analysis of the regret, we generate two different artificial i.i.d. processes to bound the probabilities related to $\bar{r}_{k,l}(t)$, $k \in \mathcal{K}_i$. The first one is the *best* process in which rewards are generated according to a bounded i.i.d. process with expected reward $\overline{\mu}_{k,l}$, the other one is the *worst* process in which the rewards are generated according to a bounded i.i.d. process with expected reward $\underline{\mu}_{k,l}$. Let $r^{\text{best}}_{k,l}(z)$ denote the sample mean of the $z$ samples from the best process and $r^{\text{worst}}_{k,l}(z)$ denote the sample mean of the $z$ samples from the worst process. We will bound the terms $E[R_n(T)]$ and $E[R_s(T)]$ by using these artificial processes along with the similarity information given in Assumption 1. The following lemma bounds $E[R_s(T)]$.

*Lemma 2:* When CoS is run with parameters $D_1(t) = t^z \log t$, $D_2(t) = F_{\max} t^z \log t$, $D_3(t) = t^z \log t$ and $m_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < 1/d$, given that $2L(\sqrt{d})^\alpha t^{-\gamma\alpha} + 6t^{-z/2} \leq a_1 t^\theta$, we have

$$\begin{aligned} E[R_s(T)] &\leq 2^{d+1}(M - 1 + |\mathcal{F}_i|)\beta_2 T^{\gamma d} \\ &\quad + 2^{d+2}(M-1)F_{\max}\beta_2 T^{\gamma d + z/2}/z \end{aligned}$$

From Lemma 2, we see that the regret increases exponentially with parameters $\gamma$ and $z$, similar to the result of Lemma 1. These two lemmas suggest that $\gamma$ and $z$ should be as small as possible, given the condition $2L(\sqrt{d})^\alpha t^{-\gamma\alpha} + 6t^{-z/2} \leq a_1 t^\theta$, is satisfied.

Each time learner $i$ selects another learner $k$ to label its data, learner $k$ calls one of its classification functions. There is a positive probability that learner $k$ will call one of its suboptimal classification functions, which implies that even if learner $k$ is near optimal for learner $i$, selecting learner $k$ may not yield a near optimal outcome. We need to take this into account, in order to bound $E[R_n(T)]$. The following lemma gives the bound on $E[R_n(T)]$.

*Lemma 3:* When CoS is run with parameters $D_1(t) = t^z \log t$, $D_2(t) = F_{\max} t^z \log t$, $D_3(t) = t^z \log t$ and $m_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < 1/d$, given that $2L(\sqrt{d})^\alpha t^{-\gamma\alpha} + 6t^{-z/2} \leq a_1 t^\theta$, we have $E[R_n(T)] \leq (2a_1 T^{1+\theta})/(1+\theta) + 4(M-1)F_{\max}\beta_2$.

From Lemma 3, we see that the regret due to near optimal arms depends exponentially on $\theta$ which is related to negative of $\gamma$ and $z$. Therefore $\gamma$ and $z$ should be chosen as large as possible to minimize the regret due to near optimal arms.

Combining the above lemmas, we obtain the finite time, uniform regret bound given in the following theorem.

*Theorem 1:* Let the CoS algorithm run with exploration control functions $D_1(t) = t^{2\alpha/(3\alpha+d)} \log t$, $D_2(t) = F_{\max} t^{2\alpha/(3\alpha+d)} \log t$, $D_3(t) = t^{2\alpha/(3\alpha+d)} \log t$ and slicing parameter $m_T = T^{1/(3\alpha+d)}$. Then,

$$\begin{aligned} E[R(T)] &\leq T^{\frac{2\alpha+d}{3\alpha+d}} \left( \frac{2(2Ld^{\alpha/2}+6)}{(2\alpha+d)/(3\alpha+d)} + 2^d Z_i \log T \right) \\ &\quad + T^{\frac{\alpha+d}{3\alpha+d}} \frac{2^{d+2}(M-1)F_{\max}\beta_2}{(2\alpha)/(3\alpha+d)} \\ &\quad + T^{\frac{d}{3\alpha+d}} 2^d (2 Z_i \beta_2 + |\mathcal{K}_i|) + 4(M-1)F_{\max}\beta_2, \end{aligned}$$

where $Z_i = \mathcal{F}_i + (M-1)(F_{\max}+1)$.

*Proof:* The highest orders of regret come from explorations and near optimal arms which are $O(T^{\gamma d + z})$ and $O(T^{1+\theta})$ respectively. We need to optimize them with respect to the constraint $2Ld^{\alpha/2} t^{-\gamma\alpha} + 6t^{-z/2} \leq a_1 t^\theta$, which is assumed in Lemmas 2 and 3. The values that minimize the regret for which this constraint hold is $\theta = -z/2$, $\gamma = z/(2\alpha)$ $a_1 = 2Ld^{\alpha/2} + 6$ and $z = 2\alpha/(3\alpha+d)$. Result follows from summing the bounds in Lemmas 1, 2 and 3. ■

*Remark 1:* Although the parameter $m_T$ of CoS depends on $T$ and hence we require $T$ as an input to the algorithm, we can make CoS run independent of the final time $T$ and achieve the same regret bound by using a well known doubling trick (see, e.g., [16]). Consider phases $\tau \in \{1, 2, \ldots\}$, where each phase has length $2^\tau$. We run a new instance of algorithm CoS at the beginning of each phase with time parameter $2^\tau$. Then the regret of this algorithm up to any time $T$ will be $O\left(T^{(2\alpha+d)/(3\alpha+d)}\right)$.

For a fixed $\alpha$, the regret becomes linear in the limit as $d$ goes to infinity. On the contrary, when $d$ is fixed the regret decreases, and in the limit as $\alpha$ goes to infinity it becomes $O(T^{2/3})$.

## *B. Regret of CoS for online learning classification functions*

In our analysis we assumed that given a context $x$, the classification function accuracy $\pi_k(x)$ is fixed. This holds when the classification functions are trained a priori, but the learners do not know the accuracy because $k$ is not tested yet. By using our contextual framework, we can also allow the classification functions to learn over time based on the data. Usually in Big Data applications we cannot have the classifiers being pre-trained as they are often deployed for the first time in a certain setting. For example in [29], Bayesian online classifiers are used for text classification and filtering. We do this by introducing time as a context, thus increasing the context dimension to $d+1$. Time is normalized in interval $[0, 1]$ such that 0 corresponds to $t = 0$, 1 corresponds to $t = T$ and each time slot is an interval of length $1/T$. For an online learning classifier, intuitively the accuracy is expected to increase with the number of samples, and thus, $\pi_k(x, t)$ will be non-decreasing in time. On the other hand, the increase in the accuracy in a single time step should be bounded. Otherwise, the online learning classifier will be able to classify all possible data streams without any error after a finite number of steps. Because of this, in addition to Assumption 1, the following should hold for an online learning classifier: $\pi_k(x, (t+1)/T) \leq \pi_k(x, t/T) + LT^{-\alpha}$, for some $L$ and $\alpha$. Then we have the following theorem when online learning classifiers are present.

*Theorem 2:* Let the CoS algorithm run with exploration control functions $D_1(t) = t^{2\alpha/(3\alpha+d+1)} \log t$, $D_2(t) = F_{\max} t^{2\alpha/(3\alpha+d+1)} \log t$, $D_3(t) = t^{2\alpha/(3\alpha+d+1)} \log t$ and slicing parameter $m_T = T^{1/(3\alpha+d+1)}$. Then,

$E[R(T)]$

$$\begin{aligned}&\leq T^{\frac{2\alpha+d+1}{3\alpha+d+1}} \left( \frac{2(2L(d+1)^{\alpha/2}+6)}{\frac{2\alpha+d+1}{3\alpha+d+1}} + 2^{d+1} Z_i \log T \right) \\ &+ T^{\frac{\alpha+d+1}{3\alpha+d+1}} \frac{2^{d+3}(M-1)F_{\max}\beta_2}{(2\alpha)/(3\alpha+d+1)} \\ &+ T^{\frac{d}{3\alpha+d+1}} 2^{d+1}(2Z_i\beta_2 + |\mathcal{K}_i|) + 4(M-1)F_{\max}\beta_2,\end{aligned}$$

where $Z_i = \mathcal{F}_i + (M-1)(F_{\max}+1)$.

## C. *Computational complexity of CoS*

For each set $P_l \in \mathcal{P}_T$, classifier $i$ keeps the sample mean of rewards from $|\mathcal{F}_i| + M - 1$ arms, while for a standard centralized bandit algorithm, the sample mean of the rewards of $| \cup_{k \in \mathcal{M}} \mathcal{F}_k|$ arms needs to be kept in memory. Since the number of sets in $\mathcal{P}_T$ is upper bounded by $2^d T^{d/(3\alpha+d)}$, the memory requirement is upper bounded by

$$(|\mathcal{F}_i| + M - 1) 2^d T^{d/(3\alpha+d)}. \qquad (2)$$

This means that the memory requirement is sublinearly increasing in $T$ and thus, in the limit $T \to \infty$ goes to infinity, however, CoS can be modified so that the available memory provides an upper bound on $m_T$. However, in this case the regret bound given in Theorem 1 may not hold.

## V. Numerical Results

In this section we provide numerical results for CoS using network security data from KDD Cup 1999 data set. We compare the performance of our learning algorithm with *AdaBoost* [30] and the online version of AdaBoost called *sliding window AdaBoost* [31]. The network security data has 42 features. The goal is to predict at any given time if an attack occurs or not. We run the simulations for three different context information; (i) context is the label at the previous time step, (ii) context is the feature named *srcbytes*, which is the number of data bytes from source to destination, (iii) context is time. All the context information is normalized to be in $[0,1]$. There are 4 local learners. Each local learner has 2 classification functions. Classification costs $d_k$ is set to 0 for all $k \in \mathcal{K}_1$.

All classification functions are trained using 5000 consecutive samples from different segments of the network security data. Then, they are tested on $T = 20000$ consecutive samples. We run simulations for two different sets of classifiers. 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 III along with the number of errors each of these classification functions made on the test data. From Table III 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 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 so they cannot simply choose

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

TABLE III

Error percentages of classification functions used by learners on the test data.

| | $D_1(t)$ | $D_2(t)$ | $D_3(t)$ | $m_T$ |
|---|---|---|---|---|
| (C1) CoS | $t^{1/8} \log t$ | $2t^{1/8} \log t$ | $t^{1/8} \log t$ | $\lceil T \rceil^{1/4}$ |
| (C2) CoS | $t^{1/2} \log t$ | $2t^{1/2} \log t$ | $t^{1/2} \log t$ | $\lceil T \rceil^{1/4}$ |

TABLE IV

Parameters for CoS

the best classification function. In all our simulations, we assume that the test data sequentially arrives to the system and the label is revealed with a one step delay.

Since we only consider single dimensional context, $d = 1$. However, due to the bursty, non-stochastic nature of the network security data we cannot find a value $\alpha$ for which Assumption 1 is true. Nevertheless, we consider two cases, C1 and C2, given in Table IV, for CoS parameter values. In C2, the parameters for CoS are selected according to Theorem 1, assuming $\alpha = 1$. In C1, the parameter values are selected in a way that will limit exploration and training. However, the regret bound in Theorem 1 may not hold for these values.

In our simulations we consider the performance of learner 1. Table V shows under each simulation and parameter setup the percentage of errors made by CoS and the percentage of time steps spent in training and exploration phases for learner 1. We compare the performance of CoS with AdaBoost and sliding window AdaBoost (SWA) whose error rates are also given in Table V. AdaBoost and SWA are trained using 20000 consecutive samples from the data set different from the test data. SWA re-trains itself in an online way using the last $w$ observations, which is called the window length. Both AdaBoost and SWA are ensemble learning methods which require learner 1 to combine the predictions of all the classification functions. Therefore, when implementing these algorithms we assume that learner 1 has access to all classification functions and their predictions, whereas when using our algorithm we assume that learner 1 only has access to its own classification functions and other learners but not their classification functions. Moreover, learner 1 is limited to use a single prediction in CoS. This may be the case in a real system when the computational capability of local learners are limited and the communication costs are high.

First, we consider the case when the parameter values are as given in C1. We observe that when the context is the previous label, CoS performs better than AdaBoost and SWA for both S1 and S2. This result shows that although CoS only uses the prediction of a single classification function, by exploiting the context information it can perform better than ensemble learning approaches which combine the predictions of all classification functions. We see that the error percentage is smallest for CoS when the context is the previous label. This is due to the bursty nature of

| (parameter, simulation) | Error % | Training % | Exploration % |
|---|---|---|---|
| (C1,S1) CoS | 0.7, 4.6, 4.8 | 0.3, 3, 2.8 | 1.4, 6.3, 8.5 |
| (C1,S2) CoS | 0.9, 39, 10 | 0.3, 3, 2.8 | 1.5, 6.5, 8.6 |
| (C2,S1) CoS | 16, 14, 41 | 8.5, 16, 79 | 55 27 20 |
| (S1, S2) AdaBoost | 4.8, 53 | | |
| (S1, S2) SWA ($w = 100$) | 2.4, 2.7 | | |
| (S2, S2) SWA ($w = 1000$) | 11, 11 | | |

TABLE V

ERROR, TRAINING EXPLORATION PERCENTAGES OF CoS FOR DIFFERENT PARAMETERS AND SIMULATIONS, AND ITS COMPARISON WITH ADABOOST AND SWA.

the attacks. When the context is the feature of the data or the time, for S1, CoS performs better than AdaBoost while SWA with window length $w = 100$ can be slightly better than CoS. But again, this difference is not due to the fact that CoS makes too many errors. It is because of the fact that CoS explores and trains other classification functions and learners. AdaBoost and SWA does not require these phases. But they require communication predictions of all classification functions and communication of all local learners with each other at each time step. Moreover, SWA re-trains itself by using the predictions and labels in its time window, which makes it computationally inefficient. Another observation is that using the feature as context is not very efficient when there are no good classifiers (S2). However, the error percentage of CoS (39%) is still lower than the error percentage of the best classifier in S2 which is 47%.

We observe that CoS performs poorly when the set of parameters is given by C2. This is due to the fact that the percentage of training and exploration phases is too large for C2, thus CoS cannot exploit the information it gathered efficiently. Another important reason for the poor performance is the short time horizon. As the time horizon grows, we expect the exploration and training rates to decrease, and the exploitation rate to increase which will improve the performance.

## VI. CONCLUSION

In this paper we developed a novel online learning algorithm for decentralized Big Data classification using context information about the high dimensional data. We proved sublinear regret results for this algorithm and showed via numerical results that it outperforms ensemble learning approaches in a network security application.

## REFERENCES

[1] 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.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] A. Slivkins, "Contextual bandits with similarity information," *arXiv preprint arXiv:0907.3986*, 2009.

[17] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," *arXiv preprint arXiv:1106.2369*, 2011.

[18] 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.

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

[20] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.

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

[22] K. Crammer and C. Gentile, "Multiclass classification with bandit feedback using adaptive regularization," 2011.

[23] A. Anandkumar, N. Michael, and A. Tang, "Opportunistic spectrum access with multiple players: Learning under competition," in *Proc. of IEEE INFOCOM*, March 2010.

[24] H. Liu, K. Liu, and Q. Zhao, "Learning in a changing world: Non-bayesian restless multi-armed bandit," *Techinal Report, UC Davis*, October 2010.

[25] C. Tekin and M. Liu, "Online learning in decentralized multi-user spectrum access with synchronized explorations," in *Proc. of IEEE MILCOM 2012*, 2012.

[26] ——, "Performance and convergence of multi-user online learning," in *2nd International ICST Conference on Game Theory for Networks (GAMENETS)*, April 2011.

[27] R. Ortner, "Exploiting similarity information in reinforcement learning," *Proceedings 2nd ICAART*, pp. 203–210, 2010.

[28] C. Tekin and M. van der Schaar, "Distributed online big data classification using context information," *arXiv preprint arXiv:1307.0781*, July 2013.

[29] K. M. A. Chai, H. L. Chieu, and H. T. Ng, "Bayesian online classifiers for text classification and filtering," in *Proc. of the 25th annual international ACM SIGIR conference on research and development in information retrieval*. ACM, 2002, pp. 97–104.

[30] 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.

[31] N. C. Oza and S. Russell, *Online ensemble learning*. University of California, Berkeley, 2001.