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

# Learning Optimal Classifier Chains for Real-time Big Data Mining

Jie Xu, Cem Tekin, and Mihaela van der Schaar
Department of Electrical Engineering
University of California, Los Angeles (UCLA)
Email: jiexu@ucla.edu, cmtkn@ucla.edu, mihaela@ee.ucla.edu

***Abstract*—A plethora of emerging Big Data applications require processing and analyzing streams of data to extract valuable information in real-time. For this, chains of classifiers which can detect various concepts need to be constructed in real-time. In this paper, we propose online distributed algorithms which can learn how to construct the optimal classifier chain in order to maximize the stream mining performance (i.e. mining accuracy minus cost) based on the dynamically-changing data characteristics. The proposed solution does not require the distributed local classifiers to exchange any information when learning at runtime. Moreover, our algorithm requires only limited feedback of the mining performance to enable the learning of the optimal classifier chain. We model the learning problem of the optimal classifier chain at run-time as a multi-player multi-armed bandit problem with limited feedback. To our best knowledge, this paper is the first that applies bandit techniques to stream mining problems. However, existing bandit algorithms are inefficient in the considered scenario due to the fact that each component classifier learns its optimal classification functions using only the aggregate overall reward without knowing its own individual reward and without exchanging information with other classifiers. We prove that the proposed algorithms achieve logarithmic learning regret uniformly over time and hence, they are order optimal. Therefore, the long-term time average performance loss tends to zero. We also design learning algorithms whose regret is linear in the number of classification functions. This is much smaller than the regret results which can be obtained using existing bandit algorithms that scale linearly in the number of classifier chains and exponentially in the number of classification functions.**

## I. Introduction

Recent years have witnessed the emergence of a plethora of online Big Data applications, such as social media analysis, video surveillance, network security monitoring and etc., which require processing and analyzing streams of raw data to extract valuable information in real-time [1]. Due to privacy issues and proprietary access to information, databases and resources [2] as well as the high computational burdens to fulfill a complex stream processing task, tasks are often decomposed into smaller pieces of subtasks which are assigned to a chain of classifiers implemented on different distributed entities, each of which being responsible for solving one subtask [3][4]. The processing results of these decomposed processing tasks are then aggregated and synthesized to fulfill desired stream mining objectives. For instance, in the video event detection problem [5], rather than directly detecting the event of interest, a set of basic concepts are first detected. The detection results of these concepts are then used to determine the presence of the event of interest. This decomposition has merits that transcend the scalability, reliability and low complexity of large-scale, real-time Big Data stream mining systems. (See Figure. 1)

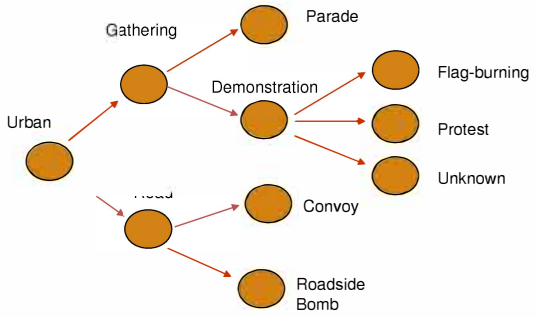



Fig. 1. An Example of Task Decomposition.

In this paper, we design online learning algorithms to optimize the classifier chains for Big Data stream mining problems. Unlike existing solutions that require the *a priori* knowledge of classification functions' performance (i.e. accuracy and cost) for various types of data characteristics, which is impossible to obtain in practice, our algorithm is able to learn their performance over time and select the optimal classifier configuration. In the considered setting, each classifier in the chain is in charge of a classification subtask (e.g. classification for a specific concept of the data) the results of which are synthesized to obtain the final classification outcome. Each component classifier maintains a number of classification functions from which it can choose the suitable one to use depending on the input data instances. The classification functions have different accuracies and costs when processing different input data sources. Importantly, these accuracies and costs are unknown and may vary over time and hence, they need to be learned online. Hence, to determine the optimal chain of classifiers, the classifiers will need to learn from past data instances and the mining performance which they obtained from a certain configuration in the past.

Learning how to design optimal classifier chains is a challenging problem [10]-[13]. First, the input data stream characteristics can be time-varying and thus, classifier chains require frequent reconfiguration to ensure acceptable classification performance. Classifier chains which are optimized offline are often not able to track the changes in the statistical

characteristics of the data over time. Therefore, an efficient solution requires learning the optimal classifier chain online. Second, classifier chains are often implemented on different distributed entities due to computational and resource constraints [10]. Joint optimization between autonomous entities can be a very difficult problem due to legal, proprietary or technical restrictions [2]. Moreover, since the data streams need to be processed in real-time, classifier chains need to minimize their end-to-end delay and thus, they need to be optimized in a distributed way without a centralized entity that can coordinate classifiers' learning process at runtime [11][13]. Third, analytics of individual classifiers may have complex relationships and hence, individual classifiers may have only limited feedback of the classification performance (e.g. the feedback information involves only the classification performance of the overall task but not the subtasks). Therefore, classifiers need to learn in a cooperative, yet distributed manner.

We model the classifier chain learning problem as a multi-player multi-armed bandit problem with limited feedback and design online learning algorithms that address all the above mentioned challenges. The proposed algorithms learn the performance of classifiers in real-time, requiring only one pass of the data set, thereby minimizing the processing delay and the memory requirements of the classifiers. The proposed algorithms do not require any runtime coordination by a central entity of distributed classifiers' learning problem and therefore, the communication overhead among classifiers is minimized. Also, our algorithms learn solely based on the mining quality of the overall task but not that of subtasks, thereby minimizing the feedback information. Most importantly, we are able to analytically bound the finite time performance of our proposed learning algorithms. Finally, we can prove that the convergence rate to the optimal classifier chain is significantly faster than that obtained by existing bandit algorithms in our considered setting.

## II. RELATED WORKS

A key research challenge [6] in Big Data stream mining systems is the management of limited system resources while providing desired application performance. Most of the existing approaches on resource-constrained stream mining problems rely on load-shedding [7][8][9], where algorithms determine a discard policy given the observed data characteristics. However, load-shedding may lead to suboptimal end-to-end performance when the data discarded at one stage is essential for a downstream (later stage) classifier. One way to overcome this limitation is to determine how the available data should be processed given the underlying resource allocation instead of to decide on which data to process, as in load-shedding based approaches. In this regard, how to optimally configure the classifier chains are extensively studied in [10]-[13]. In [10], the authors assume that the classifiers' performance is known *a priori* and determine the configuration of the multiple classifiers by solving an offline optimization problem given a fixed topology. A distributed iterative approach based on reinforcement learning techniques is proposed in [11] to optimize the classifiers' configuration given that the classifier chain topology is fixed and classifiers' performance is known. However, these offline approaches do not adapt to the dynamically changing data characteristics when the classifiers' performance is unknown and may change depending on the characteristics of the different data streams which need to be processed over time. When stream dynamics are initially unknown, a Markov model-based solution for learning the optimal rule for choosing algorithms to reconfigure the classifiers is developed in [12]. However, it requires the state information of the environment and the stream mining utility which is not available in most practical application. Moreover, for the proposed distributed implementation it does not guarantee that the optimal solution can be learned. Centralized and decentralized online solutions are proposed in [13] to tackle the problem of joint optimizing of the ordering of the chain and the configurations of classifiers. However, it also assumes that the classifiers' performance is known *a priori* and requires significant communication among classifiers. Even though different orders of classifiers can lead to varying expected processing delay when individual classifiers have the option to discard the data instances (i.e. not forward to the subsequent classifiers for processing), they do not change the mining accuracy [13]. In this paper, we focus on optimizing the classifier chain given the order and view the classifier ordering problem as an important future research topic. We rigorously design learning algorithms without *a priori* knowledge of classifiers' performance and prove that not only the learned classifier chain converges to the optimal solution but also the time-average performance loss converges to zero rapidly. This result, which, to our best knowledge, is the first in the Big Data stream mining literature, is important to ensure good mining performance during the learning process. It is especially important when the data characteristics are changing which may not allow the algorithms to determine the optimal classifier chain using *a priori* knowledge. Table I summarizes the comparison with existing works on stream mining.

We formulate the classifier chain learning problem as a multi-player multi-armed bandit problem with limited feedback. Literature on multi-armed bandit problems can be traced back to [14] which studies a Bayesian formulation and requires priors over the unknown distributions. In our paper, such infor-

| | [7][8][9] | [10] | [11][13] | [12] | |
|---|---|---|---|---|---|
| | No | Yes | Yes | Yes | Yes |
| | No | No | No | Yes | Yes |
| | Yes | No | Yes | Yes | Yes |
| | No | No | Yes | No | Yes |
| | No | Yes | No | No | Yes |

TABLE I
COMPARISON WITH EXISTING WORKS ON STREAM MINING

of all classifiers, denoted by $\tilde{\mathbf{y}}(n) = (\tilde{y}_1(n), ..., \tilde{y}_M(n))$, yields the final classification of the label $\tilde{z}(n) = \sigma(\tilde{\mathbf{y}}(n))$.

Each classifier $m$ focuses on the classification problem with respect to concept $m$ and maintains a set of classification functions $\mathcal{F}_m = \{f_{m,1}, ..., f_{m,K_m}\}$ where $f_{m,k}$ $\mathcal{X} \to \Delta(\mathcal{Y}_m)$. These classification functions can be operating points as in [10]-[13] or more sophisticated feature extraction functions, e.g. SIFT and advanced SIFT for image processing problems [25]. We also note that classifiers can be of different types. For analytical simplicity, we assume $K_m = K, \forall m$. In each period $n$, classifier $m$ picks a classification function $a_m(n) \in \mathcal{F}_m$ to classify $o(n)$. Therefore $\mathbf{a}(n) = (a_1(n), ..., a_M(n))$ represents the classifier chain that is selected in period $n$.

### B. Accuracy, cost and reward

Each classification function $f_{m,k}$ of a classifier $m$ has an unknown accuracy $\pi(f_{m,k}) \in [0, 1]$. The accuracy represents the probability that an input data instance will be classified correctly with respect to concept $m$. Calling upon a classification function $f_{m,k}$ also incurs some (communication/computation) cost. The expected cost is denoted by $d(f_{m,k})$ which is also unknown.

The accuracy $\pi(\mathbf{a})$ of a classifier chain $\mathbf{a}$ depends on the accuracies of its component classifiers. Let $\pi(\mathbf{a}) = G_\sigma(\pi(a_1), ..., \pi(a_M))$ where $G_\sigma$ depends on the deterministic function $\sigma$. The cost by calling on the classifier chain $\mathbf{a}$ is also a function of the costs of individual classifiers. Let the expected cost be $d(\mathbf{a}) = H(d(a_1), ..., d(a_M))$.

By selecting different classifier chains in different periods, the system obtains a reward that depends on the classification outcome and the incurred cost. Define the reward $r(n)$ in period $n$ as

$$r(n) = \mathbf{1}(z(n) = \tilde{z}(n)) - d(n), \tag{2}$$

where $d(n)$ is the total cost incurred in period $n$. Let the expected reward of a classifier chain $\mathbf{a}$ be $\mu(\mathbf{a}) = \pi(\mathbf{a}) - d(\mathbf{a})$.

### C. Optimal policy with complete information

The goal of the designer is to determine an algorithm $\phi$ that selects a classifier chain $\mathbf{a}(n)$ in each period $n$ that maximizes the expected reward $\mathbb{E}\{r(n)\}$. We summarize the event timeline in each period $n$ below:

- Each classifier $m$ picks a classification function $a_m(n) \in \mathcal{F}_m$ to use in this period. Hence, $\mathbf{a}(n)$ represents the classifier chain in period $n$.
- A data instance $o(n)$ enters the system and goes through the classifier chain, yielding the classified concepts $\tilde{\mathbf{y}}(n) = (\tilde{y}_1(n), ..., \tilde{y}_M(n))$. The final classification result is generated as $\tilde{z}(n) = \sigma(\tilde{\mathbf{y}}(n))$.
- At the end of the period $n$, the realized overall reward $r(n)$ according to the true label $z(n)$ and the overall cost $d(n)$ is revealed. Note that classifiers only observe this overall reward but not their own accuracies or costs.

If the accuracies and expected costs of each classification function of each classifier are known, then the optimal solution is given by (e.g. using the methods in [10][11])

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} \{\mu(\mathbf{a})\}. \tag{3}$$

That is, in each time period, the same classifier chain that maximizes the expected reward is selected. However, since the accuracies and the expected costs are unknown, the classifiers have to learn the optimal classification functions over time using the past instances and classification outcomes.

### D. The regret of learning

We define the regret as a performance measure of the learning algorithm $\phi$ used by the classifiers. Simply, the regret is the performance loss incurred due to the unknown system dynamics. Regret of a learning algorithm $\phi$ is defined with respect to the best classifier chain given in (3) and given by

$$R(n) = n\mu(\mathbf{a}^*) - \mathbb{E}\sum_{t=1}^{n} r(t). \tag{4}$$

Regret gives the convergence rate of the total expected reward of the learning algorithm to the value of the optimal solution given in (3). Any algorithm whose regret is logarithmic in time, i.e. $R(n) = O(\ln n)$, will have zero time-average regret when time goes to infinity. However, the constant that multiplies the logarithmic term $\ln n$ also has a significant impact on the time-average regret over a finite time horizon even though this impact will be eliminated when time goes to infinity. For example, suppose the optimal reward is 1, if the constant is larger than $T$, then the time average regret up to time $T$ will be larger 1 which gives too loose a bound. Therefore, this constant should be as small as possible to ensure small finite-time regret.

## IV. Learning Algorithms

In this section, we propose two distributed learning algorithms with logarithmic regret (i.e. $R(n) = O(\ln n)$). The first one applies to the general overall reward function but has a large constant that multiplies $\ln n$ (i.e $O(K^M \ln n)$). When the problem exhibits certain properties (given by Assumption 1), the second algorithm can significantly reduce the regret compared to the first one (i.e. $O(M \ln n)$).

Before we describe the learning algorithms, we introduce some notations to facilitate the analysis. We denote $\Delta_{\mathbf{a}} = \mu(\mathbf{a}^*) - \mu(\mathbf{a})$ as the difference of the expected overall reward of a classifier chain and that of the optimal classifier chain $\mathbf{a}^*$. $\mu(f_m^*, \mathbf{a}_{-m}) - \mu(f_{m,k}, \mathbf{a}_{-m})$ is the reward difference of a suboptimal classification function $f_{m,k}$ and that of the optimal classification function $f_m^*$ by classifier $m$ given the fixed choices of other classifiers $\mathbf{a}_{-m}$. We further let $\Delta_{m,k}^{min} = \min_{\mathbf{a}} \{\mu(f_m^*, \mathbf{a}_{-m}) - \mu(f_{m,k}, \mathbf{a}_{-m})\}$, $\Delta_m^{min} = \min_{f_{m,k} \neq f_m^*} \Delta_{m,k}^{min}$ and $\Delta^{min} = \min_m \Delta_m^{min}$. $\Delta^{min}$ is an important parameter which determines how accurately we can differentiate the best classifier chain and the second best classifier chain and hence, it determines the learning speed. Similarly, we can take maximum to get $\Delta_{m,k}^{max} = \max_{\mathbf{a}} \{\mu(f_m^*, \mathbf{a}_{-m}) - \mu(f_{m,k}, \mathbf{a}_{-m})\}$,

$\Delta_m^{max} = \max_{f_{m,k} \neq f_m^*} \Delta_{m,k}^{max}$ and $\Delta^{max} = \max_m \Delta_m^{max}$. $\Delta^{max}$ is important in that it determines the maximum regret (i.e. performance loss) by choosing an suboptimal classification function of a classifier in the chain. Hence, the maximum performance loss by choosing an suboptimal classifier chain is at most $M\Delta^{max}$. Finally, we denote $D = \max_{\mathbf{a}} \{\max r(n|\mathbf{a}) - \min r(n|\mathbf{a})\}$ as the bounded dynamic range of the reward for any classifier chain where $r(n|\mathbf{a})$ is the overall reward random variable given the classifier chain $\mathbf{a}$.

### A. Global learning algorithm

Since the impact of one classification function on the overall reward may vary significantly when it is cascaded with different classification functions of other classifiers, we have to explore every possible classifier chain for sufficiently many times to learn the joint impact of the classification functions in a chain on the final reward. Because the number of all possible classifier chains is large (i.e. $K^M$), this leads to a large constant that multiplies the logarithmic regret term $\ln n$.

Our algorithm consists of two phases: exploration and exploitation.

(1) *Exploration phase*: An exploration phase consists of $K^M$ periods. In these periods, all possible classifier chains are selected once in turn. This order is predetermined and hence, all classifiers know the choices of other classifiers at runtime. For example, the exploration order can be loaded into the memory of each classifier at the initialization phase of the algorithm, or classifiers can jointly decide on which exploration order to follow at the initialization phase. This allows us to minimize the communication overhead among classifiers at runtime. Let $\bar{r}(\mathbf{a})$ record the estimated average reward of the classifier chain $\mathbf{a}$. Using the realized rewards, $\bar{r}(\mathbf{a})$ is updated at the end of the exploration phase.

(2) *Exploitation phase*: The length of an exploitation phase changes over time. Each classifier maintains the number of times that it has gone through the exploration phase by the end of time $n-1$, denote by $N(n)$. Let $\zeta(n) = A \ln n$ be a deterministic function of where $A$ is a constant parameter.

- If $N(n) \leq \zeta(n)$, then the classifiers start a new exploration phase starting from time $n$.
- If $N(n) > \zeta(n)$, then each classifier $m$ selects $a_m(n)$ such that $\mathbf{a}(n) = \arg\max_{\mathbf{a}} \bar{r}(\mathbf{a})$.

The regret of this global learning algorithm is established in the following theorem.

**Theorem 1.** *If the parameter $A > 2\left(\frac{D}{\Delta^{min}}\right)^2$ , then the expected regret of the global learning algorithm after any number $n$ periods is bounded as follows*

$$R(n) \leq A(K^M - 1)M\Delta^{max} \ln n + (K^M - 1 + B)M\Delta^{max}, \tag{5}$$

*where*

$$B = \sum_{t=1}^{\infty} t^{-\frac{A}{2}\left(\frac{\Delta^{min}}{D}\right)^2}. \tag{6}$$

*Proof:* See Appendix A. ■

Theorem 1 shows that the global learning algorithm can achieve the logarithmic regret. This implies that as time $n \to \infty$, the expected time-average regret tends to 0, i.e.

$$\lim_{n \to \infty} \frac{\mathbb{E}\{R(n)\}}{n} = 0. \tag{7}$$

Note that even if a learning algorithm can learn the optimal classifier chain when time goes to infinity, it may not be able to ensure that the time-average regret is zero. Instead, our proposed algorithm guarantees that the performance loss incurred during the learning process is zero when time goes to infinity. Moreover, our learning algorithm achieves $O(K^M \ln n)$ regret which is the lowest possible regret that can be obtained. However, since the impact of individual classifiers on the overall reward can be coupled in a complex way, the algorithm has to explore every possible combination of classification functions to learn its performance. This leads to a large constant that multiplies $\ln n$ which is on the order of $K^M$.

Note that instead of using the proposed global learning algorithm, other learning algorithms such as UCB-type algorithms [16] can also be used. In this case, each classifier chain is regarded as a single arm. However, these algorithms do not improve the performance in terms of regret order, i.e. their regret is also $O(K^M \ln n)$. Moreover, they require coordinating the choices of classification functions by local classifiers at runtime which is undesirable in the distributed implementation. Since in our proposed algorithm the choices of classifier chains in the exploration phases are predetermined, classifiers do not need to exchange this information at runtime and hence, it has the advantage to reduce the communication overhead and can be more easily implemented in a distributed way.

### B. Local learning algorithm

In general the overall reward $r(\mathbf{a}$ can depend on $\mathbf{a}$ in a complex way. However, if the impact of individual classifiers on the overall reward exhibits some special property, then we can design more efficient algorithms to learn the optimal classifier chain. In this subsection, we propose an efficient learning algorithm when the following assumption holds. For instance, the overall expected reward $\mu(\mathbf{a})$ is increasing in individual expected reward $\mu(a_m) = \pi(a_m) - d(a_m)$. This is true in many practical systems [12].

**Assumption 1.** *(Monotone Contribution) There exists a function $g$ such that $\mu(\mathbf{a})$ is increasing in $g(\pi(a_m), d(a_m)), \forall m$.*

Assumption 1 implies that the optimal classification function of one classifier remains the same regardless of the choices of classification functions by other classifiers. Therefore, instead of learning the accuracy and cost of its own classification functions exactly, each classifier only needs to learn the differences between the rewards of its own classification functions, i.e., the relative reward of its own classification functions, to jointly learn how to select the optimal classifier chain.

Our new algorithm also consists of two phases: exploration and exploitation.

(1) *Exploration phase*: An exploration phase consists of $KM$ periods and is further divided into $M$ subphases with equal length of $K$ periods. Each subphase will be dedicated to the learning problem of one classifier. For classifier $\ll \cdot \emptyset$ in the $k^{th}$ period of the $i^{th}$ subphase:

- If $i = \ll \cdot \emptyset$ it chooses $a_m(n) = f_{m,k}$.
- If $i \neq \ll \cdot \emptyset$ it chooses $a_m(n) = \arg \max_{f_{m,k} \in \mathcal{F}_m} \bar{r}(f_{m,k})$.

Using the realized rewards in its own subphase, a classifier updates the reward $\bar{r}(f_{m,k}), \forall f_{m,k} \in \mathcal{F}_m$ at the end of the exploration phase.

(2) *Exploitation phase*: The length of an exploitation phase changes over time. Each classifier maintains the number of times that it has gone through the exploration phase by the end of time $n-1$, denote by $N(n)$. Let $\zeta(n) = A \ln n$ be a deterministic function of $n$ where $A$ is a constant parameter.

- If $N(n) \leq \zeta(n)$, then the classifiers start a new exploration phase starting from time $n$.
- If $N(n) > \zeta(n)$, then each classifier $\ll \emptyset$ selects $a_m(n) = \arg \max_{f_{m,k} \in \mathcal{F}_m} \bar{r}(f_{m,k})$.

Theorem 2 establishes the regret of the local learning algorithm.

**Theorem 2.** *If the parameter* $A > 2\left(\frac{D}{\Delta^{min}}\right)^2$ *, then the expected regret of the local learning algorithm after any number* $n$ *periods is bounded as follows*

$$R(n) \leq A(K-1)M\Delta^{max} \ln n + [(K-1) + 2K(M-1)e^{\frac{1}{2}\left(\frac{\Delta^{min}}{D}\right)^2} + 2B]M\Delta^{max}, \quad (8)$$

*where*

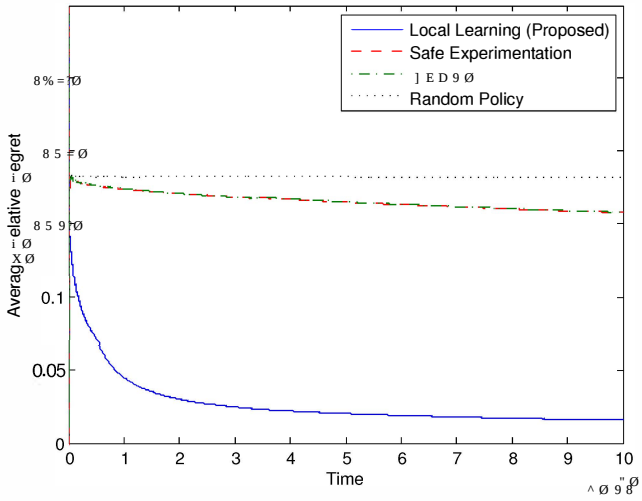
$$B = \sum_{t=1}^{\infty} t^{-\frac{A}{2}\left(\frac{\Delta^{min}}{D}\right)^2}. \quad (9)$$

*Proof:* See Appendix B. ■

Theorem 2 proves that the local learning algorithm also achieves the logarithmic regret and hence, as time $n \to \infty$, the expected time-average regret also becomes zero, i.e.

$$\lim_{\circ 1 \emptyset} \frac{\mathbb{E}\{R(n)\}}{n} = 0. \quad (10)$$

However, since Assumption 1 allows the classifiers to learn their optimal classification functions using only the relative rewards, the constant that multiplies $\ln n$ is significantly reduced in the local learning algorithm. In particular, the constant is reduced by approximately $\frac{1}{K^{M-1}}$. This improvement is enabled by the learning pattern of our proposed algorithm. In particular, even though the choices of classification functions by other classifiers may change over time and are unknown to classifier $\ll \cdot \emptyset$ they do not change within an exploration subphase for classifier $\ll 7 \emptyset$ Therefore, taking average of the realized rewards in the past still gives sufficient information of the relative reward of each classification function of classifier $\ll 7 \emptyset$ This regret result (i.e. $O(K \ln n)$) is significantly better than conventional multi-arm bandit solutions which show that the regret is linear in the number of arms (in our problem, it is the number of possible classifier chains, i.e. $K^M$).




F i g 3 .. R e g r Pe e t f o r m a Cn o o m p a r i s o n .

## V. NUMERICAL RESULTS

In this section, we provide numerical results for our proposed algorithms. We consider a problem where the objective is to determine a label $\circ(n) \in \mathbb{R}$ in each period which is a linear combination of a set of concept values. That is, $z(n) = \mathbf{h}\mathbf{y}^T(n)$ where $\mathbf{h}$ is a known coefficient vector. For instance, if $\mathbf{h} = (1/M, ..., 1/M)$, then the label $z(n)$ is the average of the concept values. Each classifier $\ll \emptyset$ maintains a set of $K$ classification functions with unknown accuracies and computation costs. It selects one of them each time to estimate $y_m(n)$. The final overall classification result $\tilde{z}(n)$ is obtained by combining the concept estimations.

Since the considered problem exhibits the monotone contribution property, we adopt the proposed local learning algorithm and compare against the widely-studied UCB1 algorithm [15] and the Safe Experimentation learning algorithm adopted in [10][11]. To provide the worst case performance, we also implement a random policy which randomly selects a classifier chain in each period. Figure 3 shows the average regret of these four algorithms when $M = 4, K = 3$. The curves are generated by averaging over 100 experiments. Since the UCB1 treats every classifier chain as an arm, the convergence speed is very slow. Because the Safe Experimentation algorithm requires the accurate knowledge of classification functions' performance, it performs poorly when such information is available in the considered setting. By exploiting the monotone contribution property, the local learning algorithm significantly outperforms the safe experimentation algorithm and UCB1 in terms of much lower regret (performance loss). In Table III, we further show the time-average relative regret of UCB1 and the proposed local learning algorithm after $10^5$ periods for varying number of classifiers. The number of classification functions of each classifier is fixed to be $K = 3$. As we can see, the

| | M = 1 | M = 2 | M = 3 | M = 4 | M = 5 |
|---|---|---|---|---|---|
| UCB1 | 0.0026 | 0.0148 | 0.0546 | 0.1249 | 0.1548 |
| Proposed | 0.0025 | 0.0030 | 0.0031 | 0.0073 | 0.0166 |

TABLE III
REGRET PERFORMANCE FOR VARYING NUMBER OF CLASSIFIERS.

| | UCB1 | Local learning |
|---|---|---|
| Message exchange | $O(M)$ | 0 |
| Memory requirement | $O(K^M)$ | $O(KM)$ |
| Regret | $O(K^M \ln n)$ | $O(K \ln n)$ |

TABLE IV
IMPLEMENTATION COMPLEXITY AND REGRET COMPARISON.

performance gain increases significantly with the increase of the number of classifiers since the arm space of the UCB1 algorithm has to increase exponentially with the $M$ thereby leading to a very slow convergence rate when $M$ is large. Finally, Table. IV compares the implementation complexity and learning regret of UCB1 and the local learning algorithm.

## VI. CONCLUSION

In this paper, we proposed online distributed algorithms to learn the optimal classifier chains for real-time Big Data stream mining applications. To our best knowledge, this paper is the first that formulates this problem as a bandit problem and provides the first analytical results on the learning performance for such Big Data applications. The learning regret of the proposed algorithm is linear in the number of classification functions which is much smaller than the regret results that can be obtained by existing bandit algorithms that scale linearly in the number of classifier chains (and hence, it scales exponentially in the number of classification functions).

## APPENDIX A
## PROOF OF THEOREM 1

We first prove that after $L$ exploration phases, the probability that a non-optimal classifier chain $\mathbf{a} \neq \mathbf{a}^*$ is selected in an exploitation slot is at most $2e^{-\frac{L}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2}$. A non-optimal classifier $\mathbf{a} \neq \mathbf{a}^*$ is selected in an exploitation slot only if $\bar{r}(\mathbf{a}) > \bar{r}(\mathbf{a}^*)$. Notice that

$$\begin{aligned} & P(\bar{r}(\mathbf{a}) < \bar{r}(\mathbf{a}^*)) \\ & > P(\bar{r}(\mathbf{a}^*) > \mu(\mathbf{a}^*) - 0.5\Delta_{\mathbf{a}}) \cdot P(\bar{r}(\mathbf{a}) < \mu(\mathbf{a}) + 0.5\Delta_{\mathbf{a}}). \end{aligned} \tag{11}$$

Therefore,

$$\begin{aligned} & P(\bar{r}(\mathbf{a}) > \bar{r}(\mathbf{a}^*)) = 1 - P(\bar{r}(\mathbf{a}) < \bar{r}(\mathbf{a}^*)) \\ < \;& 1 - P(\bar{r}(\mathbf{a}^*) > \mu(\mathbf{a}^*) - 0.5\Delta_{\mathbf{a}}) \\ & \quad \cdot P(\bar{r}(\mathbf{a}) < \mu(\mathbf{a}) + 0.5\Delta_{\mathbf{a}}) \\ = \;& 1 - (1 - P(\bar{r}(\mathbf{a}^*) < \mu(\mathbf{a}^*) - 0.5\Delta_{\mathbf{a}})) \\ & \quad \cdot (1 - P(\bar{r}(\mathbf{a}) > \mu(\mathbf{a}) + 0.5\Delta_{\mathbf{a}})) \\ < \;& P(\bar{r}(\mathbf{a}^*) < \mu(\mathbf{a}^*) - 0.5\Delta_{\mathbf{a}}) + P(\bar{r}(\mathbf{a}) > \mu(\mathbf{a}) + 0.5\Delta_{\mathbf{a}}). \end{aligned} \tag{12}$$

By Hoeffding's inequality,

$$\begin{aligned} & P(\bar{r}(\mathbf{a}^*) < \mu(\mathbf{a}^*) - 0.5\Delta_{\mathbf{a}}) \\ & = P(\bar{r}(\mathbf{a}) > \mu(\mathbf{a}) + 0.5\Delta_{\mathbf{a}}) < e^{-\frac{L}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2}. \end{aligned} \tag{13}$$

Therefore,

$$P(\bar{r}(\mathbf{a}) > \bar{r}(\mathbf{a}^*)) < 2e^{-\frac{L}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2}. \tag{14}$$

By construction, at any time $n$, at most $\zeta(n)+1$ exploration phases have been experienced. Hence, at most $K^M(\zeta(n)+1)$ exploration slots have been experienced. For these slots, the regret is at most

$$(\zeta(n)+1)\sum_{\mathbf{a}\neq\mathbf{a}^*}\Delta_{\mathbf{a}} = A\sum_{\mathbf{a}\neq\mathbf{a}^*}\Delta_{\mathbf{a}}\ln n + \sum_{\mathbf{a}\neq\mathbf{a}^*}\Delta_{\mathbf{a}} \tag{15}$$

$$\leq A(K^M-1)M\Delta^{max}\ln n + (K^M-1)M\Delta^{max}. \tag{16}$$

At any time $t < n$ when it is an exploitation slot, the expected regret by choosing a non-optimal classifier chain $\mathbf{a} \neq \mathbf{a}^*$ is at most

$$2\Delta_{\mathbf{a}}e^{-\frac{1}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2\zeta(t)} < 2\Delta_{\mathbf{a}}e^{-\frac{1}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2 A\ln n} = 2\Delta_{\mathbf{a}}t^{-\frac{A}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2}. \tag{17}$$

The expected regret in the exploitation slots up to time $n$ is at most

$$\begin{aligned} & \sum_{t=1}^{n} 2\max_{\mathbf{a}}\Delta_{\mathbf{a}}t^{-\frac{A}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2} < 2M\Delta^{max}\sum_{t=1}^{n} t^{-\frac{A}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2} \\ & 2M\Delta^{max}\sum_{t=1}^{L\zeta(n)} t^{-\frac{A}{2}\left(\frac{\Delta_{\mathbf{a}}}{D}\right)^2} < 2BM\Delta^{max}. \end{aligned} \tag{18}$$

Combining the regret in the exploration periods and the exploitation periods, the total expected regret is at most

$$A(K^M-1)M\Delta^{max}\ln n + (K^M-1+B)M\Delta^{max} \tag{19}$$

## APPENDIX B
## PROOF OF THEOREM 2

**Lemma 1.** *After exploration phases, the probability that a non-optimal classification function $f_{m,k} \neq f_m^*$ is selected by classifier $c_l$ is at most $2e^{-\frac{L}{2}\left(\frac{\Delta_{m,k}^{min}}{D}\right)^2}$*

*Proof:* A non-optimal classification function $f_{m,k} \neq f_m^*$ is selected if $\bar{r}(f_{m,k}) > \bar{r}(f_m^*)$. Let $\mathbf{b}_{-m}^l$ be the classification functions selected by other classifiers at the $l^{th}$ exploration phase, then

$$\mathbb{E}\bar{r}(f_{m,k}) = \frac{1}{L}\sum_{l=1}^{L}\mu(f_{m,k}, \mathbf{b}_{-m}^l). \tag{20}$$

Since $\mu(f_m^*, \mathbf{b}_{-m}) - \mu(f_{m,k}, \mathbf{b}_{-m}) \geq \Delta_{m,k}^{min}, \forall \mathbf{b}_{-m}$,

(21)

Because the realized reward is bounded, according to Hoeffding's inequality, we have

$$P(\bar{r}(f_{m,k}) > \bar{r}(f_m^*)) < 2e^{-\frac{L}{2}\left(\frac{\Delta_{m,k}^{min}}{D}\right)^2}. \tag{22}$$

■

*Proof of Theorem 2*: At any time $n$, at most $\zeta(n)+1$ exploration phases have been experienced. In the $l^{th}$ exploration phase and the $m^{th}$ subphase, the regret caused by classifier $m$ is at most $\sum_{f_{m,k}\neq f_m^*} \Delta_{m,k}^{max}$. By Lemma 1, the regret caused by classifier $i \neq m$ is at most

$$K\Delta_{\$\emptyset}^{max} 2e^{(\ )\,\mathrm{P}\,\emptyset\left(\frac{\Delta_i^{min}}{D}\right)^2}. \tag{23}$$

The regret caused in the exploration phases up to the $L^{th}$ exploration phase is at most,

$$\begin{aligned}
&\sum_{l=1}^{L}\sum_{m=1}^{M}\left(\sum_{f_{m,k}\neq f_m^*}\Delta_{m,k}^{max} + \sum_{i\neq m} K_{\ i}^{\ max} e^{\frac{l-1}{2}\left(\frac{\Delta_i^{min}}{D}\right)^2}\right)\\
&< L(K-1)M\Delta^{max}\\
&\quad + \sum_{l=1}^{\infty} K(M-1)M\Delta^{max} 2e^{-\frac{l-1}{2}\left(\frac{\Delta_i^{min}}{D}\right)^2}\\
&< L(K-1)M\Delta^{max} + 2K(M-1)M\Delta^{max} e^{\frac{1}{2}\left(\frac{\Delta_i^{min}}{D}\right)^2}.
\end{aligned} \tag{24}$$

Since at any time $n$, at most $\zeta(n)+1$ exploration phases have been experienced, the regret in the exploration phases is at most

$$\begin{aligned}
&A(K-1)M\Delta^{max}\ln n\\
&+[(K-1)+2K(M-1)e^{\frac{1}{2}\left(\frac{\Delta_i^{min}}{D}\right)^2}]M\Delta^{max}.
\end{aligned} \tag{25}$$

At any time $t < n$ when it is an exploitation period, the expected regret by choosing a non-optimal classification function $f_{m,k} \neq f_m^*$ for classifier $m$ is at most (by Lemma 1)

$$\begin{aligned}
&2\Delta_{m,k}^{max} e^{\mathrm{a}\frac{\zeta(n)}{\emptyset\,2}\left(\frac{\Delta_{m,k}^{min}}{\mathrm{J}\emptyset}\right)^2}\\
&< 2\Delta_{m,k}^{max} e^{\frac{\{\emptyset\,\bar{0}\,\bar{n}\,e}{2}\left(\frac{\Delta_{m,k}^{min}}{D}\right)^2} = 2\Delta_{m,k}^{max} t^{\cdot^{\{\emptyset}_{<}\left(\frac{\Delta_{m,k}^{min}}{D}\right)^2}}.
\end{aligned} \tag{26}$$

The expected regret in the exploitation slots up to time $n$ is at most

$$\begin{aligned}
&\sum_{t=1}^{n}\sum_{m=1}^{M} 2\Delta^{max} t^{-\frac{A}{2}\left(\frac{\Delta_{m,k}^{min}}{D}\right)^2} = 2M\Delta^{max}\sum_{t=1}^{n} t^{-\frac{A}{2}\left(\frac{\Delta_{m,k}^{min}}{D}\right)^2}\\
&2M\Delta^{max}\sum_{t=1}^{\infty} t^{-\frac{A}{2}\left(\frac{\Delta_i^{min}}{D}\right)^2} < 2BM\Delta^{max}.
\end{aligned} \tag{27}$$

Combining the regret in the exploration periods and exploitation periods, the total regret is at most

$$\begin{aligned}
&A(K-1)M\Delta^{max}\ln n\\
&+[(K-1)+2K(M-1)e^{\frac{1}{2}\left(\frac{\Delta^{min}}{D}\right)^2}+2B]M\Delta^{max}.
\end{aligned} \tag{28}$$

## REFERENCES


[1] M. Shah, U. Srivastava, Nina Taft, Ramakrishnan, and Franklin, "Dynamic optimization of continuous queries over sensor networks," *Proc. IEEE Int. Conf. Data Engineering (ICDE)*, 2003.

[2] S. Merugu and Ghosh, "Privacy-preserving distributed clustering using generative models," *Proc. Int. Conf. Data Mining (ICDM)*, 2003.

[3] M. Cherniack, Balakrishnan, Christian, Cetintemel, Franklin, and S. Zdonik, "Scalable distributed stream processing," *Proc., Conf. Innovative Data Systems Research (CIDR)*, 2003.

[4] O. Madani and Widom, "Adaptive filters over continuous queries over distributed data streams," *ACM SIGMOD 2003*.

[5] Y. Jiang, Bhattacharya, Chang, and M. Shah, "High-level event recognition in unconstrained videos," *Multimed. Info. Retr.*, Nov. 2013.

[6] M. Gaber, Zaslavsky, Krishnaswamy, "Resource-aware knowledge discovery in data streams," *Proc. 1st Int. Workshop on Knowledge Discovery in Data Streams*, 2004.

[7] B. Babcock, Babu, Datar, Motwani, "Chain: operator scheduling for memory minimization in data stream systems," *ACM SIGMOD*, 2003.

[8] N. Tatbul, Cetintemel, Zdonik, Cherniack, Stonebraker, "Load shedding in a data stream manager," *Proc. 29th Int. Conf. Very Large Data Bases (VLDB)*, 2003.

[9] Y. Chi, P. S. Yu, H. Wang, R. R. Muntz, "Loadstar: load shedding in data stream mining," *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2004.

[10] F. Fu, D. Turaga, Verscheure, M. van der Schaar, "Configuring competing multimedia data streams over limited distributed computing platforms," *IEEE, Journal of Selected Topics in Signal Processing*, vol. 1, no. 2, Dec. 2007.

[11] F. Fu and M. van der Schaar, "Adaptive operation of decentralized dynamic control systems over heterogeneous distributed platforms," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 3035–3048, Nov. 2010.

[12] F. Fu and M. van der Schaar, "Rules-of-thumb based configuration of multi-class multi-hop real-time multimedia systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, Article ID 975640, 2009.

[13] D. Turaga, Chen, and M. van der Schaar, "Adaptive topologic optimization for large-scale stream mining," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 3, June 2010.

[14] Whittle, "Multi-armed bandits and the Gittins index," *Journal of the Royal Statistical Society, Series B (Methodological)*, 1980.

[15] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, 1985.

[16] P. Auer, N. Cesa-Bianchi, Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2-3, 2002.

[17] V. Anantharam, Varaiya, and J. Walrand, "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part I: I.I.D. rewards," *IEEE Trans. Autom. Control*, vol. AC-32, no. 11, 1987.

[18] A. Anandkumar, Michael, Tang, and A. Swami, "Distributed algorithms for learning and cognitive medium access with logarithmic regret," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 4, Apr. 2011.

[19] Gai, Krishnamachari, Jain, "Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Networking*, vol. 20, no. 5, 2012.

[20] Rusmevichientong and Tsitsiklis, "Linearly parameterized bandits," *Math. Oper. Res.*, vol. 35, no. 2, 2010.

[21] Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, 2002.

[22] Dani, Hayes, and Kakade, "Stochastic linear optimization under bandit feedback," *Proc. 21st Annu. COLT*, 2008.

[23] Tekin and M. Liu, "Online Learning of Rested and Restless Bandits," *IEEE Trans. on Information Theory*, 58, pp. 5588–5611, 2012.

[24] H. Liu, K. Liu, and Q. Zhao, "Learning in a changing world: restless multi-armed bandit with unknown dynamics," *IEEE Trans. on Information Theory*, 59, pp. 1902–1916, 2013.

[25] Sidiropoulos, Mehrotra, Liko, "Emphasis, viewing and concept detection in video of topographical maps," *Proc. IEEE International Conference on Image Processing (ICIP)*, 2013.