Review

ENCYCLOPEDIA WITH SEMANTIC COMPUTING AND ROBOTIC INTELLIGENCE
Vol. 1, No. 1 (2017) 1630011 (8 pages)

DOI: 10.1142/S2425038416300111

World Scientific
www.worldscientific.com

Actionable intelligence and online learning for semantic computing

Cem Tekin*,‡ and Mihaela van der Schaar†,§
*Electrical and Electronics Engineering Department
Bilkent University, Ankara, Turkey
†Electrical Engineering Department, University of California
Los Angeles, California, USA
‡cemtekin@ee.bilkent.edu.tr
§mihaela@ee.ucla.edu

Accepted 13 September 2016; Published 17 March 2017

As the world becomes more connected and instrumented, high dimensional, heterogeneous and time-varying data streams are collected and need to be analyzed on the fly to extract the actionable intelligence from the data streams and make timely decisions based on this knowledge. This requires that appropriate classifiers are invoked to process the incoming streams and find the relevant knowledge. Thus, a key challenge becomes choosing online, at run-time, which classifier should be deployed to make the best possible predictions on the incoming streams. In this paper, we survey a class of methods capable to perform online learning in stream-based semantic computing tasks: multi-armed bandits (MABs). Adopting MABs for stream mining poses, numerous new challenges requires many new innovations. Most importantly, the MABs will need to explicitly consider and track online the time-varying characteristics of the data streams and to learn fast what is the relevant information out of the vast, heterogeneous and possibly highly dimensional data streams. In this paper, we discuss contextual MAB methods, which use similarities in context (meta-data) information to make decisions, and discuss their advantages when applied to stream mining for semantic computing. These methods can be adapted to discover in real-time the relevant contexts guiding the stream mining decisions, and tract the best classifier in presence of concept drift. Moreover, we also discuss how stream mining of multiple data sources can be performed by deploying cooperative MAB solutions and ensemble learning. We conclude the paper by discussing the numerous other advantages of MABs that will benefit semantic computing applications.

Keywords: Stream mining; online learning; multi-armed bandits; semantic computing.

1. Introduction

Huge amounts of data streams are now being produced by more and more sources and in increasingly diverse formats: sensor readings, physiological measurements, GPS events, network traffic information, documents, emails, transactions, tweets, audio files, videos etc. These streams are then mined in real-time to assist numerous semantic computing applications (see Fig. 1): patient monitoring,[1] personalized diagnosis,[2,3] personalized treatment recommendation,[4,5] recommender systems,[6–8] social networks,[9] network security,[10] multimedia content aggregation,[11] personalized education[12,13] etc. Hence, online data mining systems have emerged that enable such applications to analyze, extract actionable intelligence and make decisions in real-time, based on the correlated, high-dimensional and dynamic data captured by multiple heterogeneous data sources. To mine the data streams, the following questions need to be answered continuously: Which classifiers should process the data? How many and in which order, configuration or topology? What are the costs (e.g., delay) and benefits (accuracy of predictions) in invoking a specific classifier?

In this paper, we formalize the real-time mining of data streams for the purpose of semantic computing as an online learning and sequential decision problem, where classifiers and their configurations are chosen online to make predictions based on the gathered data, and subsequently focus on multi-armed bandits (MABs) as an important class of solutions for solving this problem. In the considered systems, such as the example given in Fig. 2, data from multiple sources are processed by a learner which determines on-the-fly how to classify the different data streams and make decisions based on the predictions. For this, the learner uses one of its available classifiers (or an ensemble of classifiers or classifier chains) to make a prediction. Since the prediction accuracy of the classifiers changes dynamically, over time, based on the characteristics of the collected data streams, this needs to be learned online.[14] Hence, the learner needs to continuously learn while at the same time make accurate predictions and decisions, i.e., the learning and decision making are coupled and concurrent.

Such online learning and sequential decision making under uncertainty problems can be modeled as MAB.[15,16] In the MAB framework, the learner chooses actions (bandits' arms) at each time slot and, based on this, random rewards (feedback) are revealed. Previously, MAB methods are applied to solve problems in clinical trials,[15] multi-user communication networks[17] and recommender systems.[18,19] A key advantage of MABs as compared to other online learning

Fig. 1. Semantic computing applications.

methods (e.g., see Refs. 20 and 21) is that they can provide a bound on the convergence speed as well as a bound on the loss due to learning compared to an oracle solution which requires knowledge of the stochastic model of the system, which is named regret. In stream mining, regret of a learner's algorithm at time T is defined as the difference between the expected number of correct predictions minus costs of making a prediction using this algorithm and the expected number of correct predictions minus costs of the "oracle" algorithm which acts optimally by knowing the accuracies of all classifiers for all data and contexts. The regret, which is a non-decreasing function of T, is denoted by $\mathrm{Reg}(T)$. Any algorithm whose regret grows sublinearly in T will acquire the same average reward with the "oracle" algorithm as $T \to \infty$.

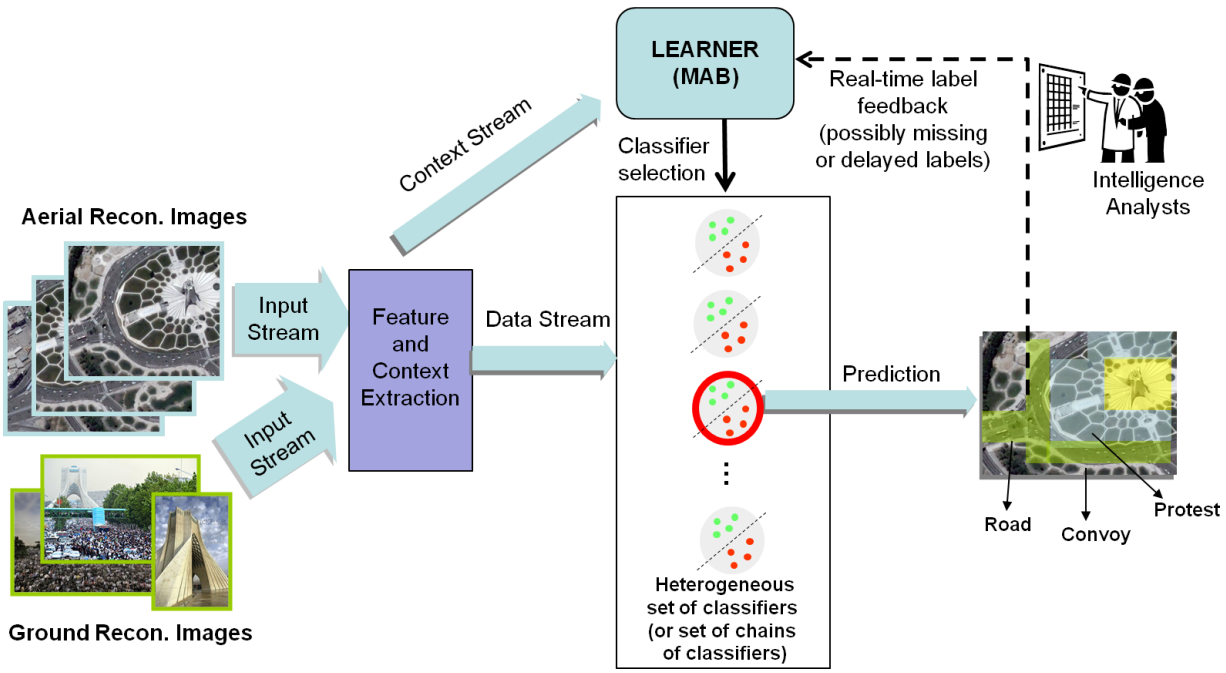

Fig. 2. Online learning of the best classifier (or the best chain of classifiers) to maximize the prediction accuracy, based on the data and context stream characteristics, through predictions and label feedback which can be delayed or missing.

Importantly, unlike many of the aforementioned MAB works, the focus in stream mining is on making decisions based on data streams ("data-in-motion") with different contexts, rather than based on channel or network states, user feedbacks or recommendations, etc. Therefore, applying MABs to formalize and solve stream mining problems presents numerous new challenges which will be discussed in the subsequent sections. To address these new and unique challenges, we will focus on a special class of MABs, referred to as contextual MABs,[22,23] which are especially suitable because they can exploit the (automatically) generated meta-data, i.e., the context, which is gathered or associated to the data streams in the process of capturing or preprocessing them (e.g., location, data modality etc.), or the features extracted directly from the data during a pre-processing phase. Since it is unknown *a priori* which contexts should be exploited at each moment in time to yield the best predictions based on the time-varying data characteristics, it becomes essential for the efficiency of contextual MABs when applied to stream mining to find the relevant contexts on the fly.[8]

2. Formalizing Real-Time Stream Mining Problems as MABs

In a stream mining system, the learner is equipped with a set of classifiers $\mathcal{F}$, each of which provides a prediction when called by the learner, based on the data. Time varying and heterogeneous characteristics of data streams makes it impossible to design a classifier which works well for any data. Usually, the classifiers are specialized on specific data streams, for which they can produce accurate predictions. Since the characteristics of the data stream is unknown *a priori*, the learner needs to learn which classifiers to choose online. Classifiers used by the learner can be pre-trained or they can even learn online based on the predictions they make on the data and the labels they receive. Choosing the right classifier for a given context is a highly nontrivial problem due to lack of *a priori* knowledge about performance of the classifiers on the dynamic and heterogeneous data as well as the lack of *a priori* knowledge about the relationships between the contexts and prediction accuracies.

An important goal in stream mining is to balance the short term and long term performance. Stream applications run indefinitely, and there is no predetermined final time T for which the learner can plan an optimal learning strategy. The learner's reward at time t is defined as the prediction accuracy of the classifier chosen at time t minus costs of prediction. Therefore, the learner should continuously balance exploration, i.e., trying different classifiers to estimate their accuracies, and exploitation, i.e., selecting the classifier with the highest estimated reward to maximize the instantaneous reward, to have high expected total reward at any time slot. Without loss of generality, in our discussion we assume that costs of selecting classifiers is the same for each classifier, thus we focus on maximizing the number of correct predictions.

We first explain how the data, labels and contexts are generated. It is assumed that at each time slot $t = 1, 2, \ldots$, data $s(t)$, label $y(t)$ and context $\boldsymbol{x}(t)$ are drawn from an unknown joint distribution J over $\mathcal{S} \times \mathcal{Y} \times \mathcal{X}$, which is called the *stream distribution*, where $\mathcal{S}$ is the set of data instances, $\mathcal{Y}$ is the set of labels and $\mathcal{X}$ is the set/space of contexts. It is usually assumed that $\mathcal{X}$ is very large.[18,22] The conditional distribution of data and label given context $\boldsymbol{x}$ is $G_{\boldsymbol{x}}$ on $\mathcal{S} \times \mathcal{Y}$, and it depends on J. The accuracy of classifier f for context $\boldsymbol{x}$ is given by $\pi_f(\boldsymbol{x})$. Hence, the optimal classifier given context $\boldsymbol{x}$ is $f^*(\boldsymbol{x}) := \arg\max_{f \in \mathcal{F}} \pi_f(\boldsymbol{x})$. Classifier accuracies are unknown to the learner since J is unknown. Learning J from past observations and using the prediction rules of the classifiers to estimate the classifier accuracies is not feasible both because of the dimensionality of the data stream and computational issues related to the complexity of the classification rules. In order to overcome this issue, MAB methods directly learn the accuracy without estimating J.

Data streams usually have the *similarity property*,[22] which implies that the prediction accuracies of a classifier for two contexts $\boldsymbol{x}$ and $\boldsymbol{x}'$ are related to each other when the contexts are related to each other. There are many metrics to define similarity between contexts and similarity between classifier accuracies, and one of the most widely used metric is the Hölder continuity metric, which is defined as

$$|\pi_f(\boldsymbol{x}) - \pi_f(\boldsymbol{x})| \leq L||\boldsymbol{x} - \boldsymbol{x}'||^{\gamma} \tag{1}$$

for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$, where $||\cdot||$ is the standard Euclidian metric, and $L > 0$ and $\gamma > 0$ are constants that define the structure of similarity. Some MAB methods learn independently for each context,[16] while some MAB methods exploit the similarity between classifier accuracies.[22]

While in the above formulation the stream distribution is static, numerous works considered time-varying distributions for classification rules,[11] which is also called *concept drift*.[24,25] A change in accuracy of a classifier can happen when the classifier updates its prediction rule. For example, online learning classifiers can update their prediction rule based on the past predictions and labels, and hence can improve their accuracy. In this case, the prediction rule of a classifier will be time varying. Therefore, a more systematic characterization of concept drift is *accuracy drift*, i.e., the change in classifier accuracies which can be the result of concept drift, prediction rule update or both. An important class of this is gradual accuracy drift, i.e., for each $t, t' \in \{1, 2, \ldots\}$ and $f \in \mathcal{F}$, there exists constants $\gamma > 0$ and $L > 0$ such that for all $\boldsymbol{x}, \boldsymbol{x}'$, we have

$$|\pi_{f,t}(\boldsymbol{x}) - \pi_{f,t'}(\boldsymbol{x}')| \leq L(||\boldsymbol{x} - \boldsymbol{x}'||^2 + |t'/\tau - t/\tau|^2)^{\gamma/2} \tag{2}$$

where $\pi_{f,t}(\cdot)$ is the time-varying accuracy of classifier f, τ is the *stability of accuracy*, hence $1/\tau$ is the *speed of drift*. This equation limits how much the similarity between the accuracies of classifier f for two contexts can change with time. Knowing this, the learner can analytically determine the window of history of past predictions, labels and decisions it should take into account when estimating the current classifier accuracies.[11] Intuitively, as τ increases, the learner can rely on a larger window of history to estimate the classifier accuracies. MAB methods can also be used for the case when the accuracy drift is abrupt but infrequent, i.e., there can be a large change in a classifier's accuracy between two consecutive time slots, but the frequency of such changes is low so that the changes can be tracked.[26] Thus, the important design challenge is how to optimally combine the past observations to estimate classifier accuracies. In the following sections, we will review several methods for this.

3. Dealing with Heterogeneous and Dynamic Data: The Role of Contextual MABs

The idea behind contextual bandit algorithms is to form a partition of the context space consisting of sets of contexts such that the classifier accuracies are estimated independently for each set instead of being estimated independently for each context, based on the *mining history*.[19,22] The partition of the context space can be even generated on the fly, based on the past context arrivals and the similarity information in Eq. (1) given to the learner. The number of past contexts that lie in a set as well as the variation of classifier accuracies for contexts within that set, increases with the size (volume) of the set. From an estimation perspective, the first one can be seen as an increase in the sample size, and the second one can be seen as a decrease in the sample quality. An optimal contextual bandit algorithm should balance the tradeoff between these two. Since accuracy drift in Eq. (2) translates into an increase in the variation of classifier accuracies between past time slots and the current time slot, the partitioning idea of contextual bandits also works well in this setting.

To differentiate between different types of contexts, we write the context (vector) at time step t as $\boldsymbol{x}(t) = (x_1(t), \ldots, x_d(t))$, where d is the dimension of the context vector and $x_i(t)$ is a type-i context that lies in the set of type-i contexts $\mathcal{X}_i$, which can be either discrete or continuous. Hence, $\mathcal{X}$ is equal to the Cartesian product of type-i context sets. For a data stream, one type of context can be the packet size, while another one can be the location. Even a feature of the raw data, or the output of a preprocessor on data can be a type of context. There are two ways to partition the contexts. The first way, i.e., the *non-adaptive contexts* method, is to form a partition over $\mathcal{X}$ to estimate the classifier accuracies for each set in that partition without learning about the impact of different types of contexts in the context vector to the accuracy of classifier selection.[2,7,18,22] The second way, i.e., the *adaptive contexts* method, is to form partitions over the context space by learning the *relevant contexts*, i.e., the contexts whose values affect the prediction accuracy the most.[8] The adaptive contexts method learns the classifier accuracies much faster, especially when the number of relevant context types is small. However, it requires more processing power, since it must handle the task of identifying the relevant contexts in addition to choosing the best classifier. Nonadaptive contexts method has been successfully applied in numerous semantic computing applications including context-driven image stream mining[2] and item recommendation in social networks.[9] Similarly, adaptive contexts methods has been successfully applied in context-driven medical expertise discovery,[4] multimedia content aggregation[11] and recommender systems.[8]

In the presence of accuracy drift, the learner needs to choose classifiers to make predictions solely on the recent relevant history. An important challenge arises due to the fact the learner cannot trust on the entire mining history, because past predictions, labels and classifier selection may have become irrelevant due to the concept drift or change in classifiers. Thus, the learner should decide which parts of the mining history, it should use to estimate the classifier accuracies to balance the estimation errors due to small number of past samples and variations in the classifier accuracies over time. Hence, the number of past samples that should be used when estimating the accuracies should be optimized depending on the speed of the drift, i.e., $1/\tau$ in Eq. (2).

Approaches used in dealing with accuracy drift include using a sliding time window of mining history to estimate accuracies or combining time slots into rounds where a new MAB-based learning algorithm run for each round.[26] When using a sliding time window, creating the correct partition of the context space becomes complicated since the sets in the partition could also merge to form bigger sets when past mining histories are discarded in forming accuracy estimates. The second approach creates the problem of cold-start, which slows down learning and increases loss due to explorations, since mining history in the previous rounds is completely neglected when deciding how to choose classifiers in the initial slots of the current round. An alternative to both of these methods is to run different instances of the MAB algorithm with overlapping rounds, where each round is a time window.[11] In this method, time is divided into rounds with multiple decision epochs. Each round is further divided into two passive and active sub-rounds of equal lengths such that active sub-round of round $\rho - 1$ overlaps with the passive sub-round of round ρ. At the beginning of round ρ a new instance of the MAB algorithm, denoted by MAB_ρ, is created. In the passive sub-round, MAB_ρ learn from the classifier selections made by $\text{MAB}_{\rho-1}$ but does not select any classifier. In the active sub-round, MAB_ρ both learn from the predictions and labels and also selects the classifier to produce the prediction.

4. Learning from Multiple Data Sources: Cooperative MABs and Ensemble of MABs

4.1. *Cooperative MABs*

Cooperative stream mining emerged as a result of applications running in geographically different locations such as security cameras, distributed databases of large companies, cloud servers, etc., as well as applications running under different software platforms and applications running by different entities/companies. In this section, we review decentralized and cooperative learning implementations named cooperative contextual bandits.[27] In these problems, the data stream of each source is locally processed by a learner which has its own set of classifiers. When data arrives along with the context, a learner can use one of its own classifiers to make a prediction, or it can call another learner and ask for a prediction. When calling another learner, the learner can send both its data and context (high cost, better prediction) or just its context (low cost, worse prediction). The goal of each learner is to maximize its total expected reward (correct predictions minus cost) by learning the best distributed context-dependent classification strategy.

Cooperation by learners not knowing other learners expertise (i.e., the classifiers used by other learners and their accuracies when processing a specific type of data) requires a novel 3-phase learning structure involving training, exploration and exploitation phases as compared to the conventional single-learner solutions. The additional training phase for learner i serves the purpose of helping other learners discover their best classifiers for learner i's contexts. Since the data and context arrivals to learners are different, without the training phase, learner j may not be able to learn its best action for learner i's contexts, hence learner i may not be able to assess the potential of learner j in making predictions about i's data.

Cooperative contextual bandits have been successfully applied in applications with decentralized learners. In one application, the problem of assigning patients to medical experts at different clinics is considered.[4] The system model for this application is given in Fig. 3. In this application, clinics, that act as decentralized entities, manage to learn to match each new patient with the best expert for that patient via cooperation. Experts in this application can be viewed as classifiers. It is assumed that a clinic knows its own experts but does not know the experts available at other clinics. However, clinics are not aware of the quality/accuracy of their own experts and of other clinics. Moreover, unlike the standard one-size-fits-all approach in which the new patient is matched with the best expert on average, the goal in this application is to provide personalized matching based on the context $\boldsymbol{x}$ of the new patient. For instance, some senior experts may be substantially better at diagnosing patients with complex co-morbidities than junior experts. In addition, the diagnostic accuracy of an expert may depend on family history, age, weight or genes of the patient. All of these

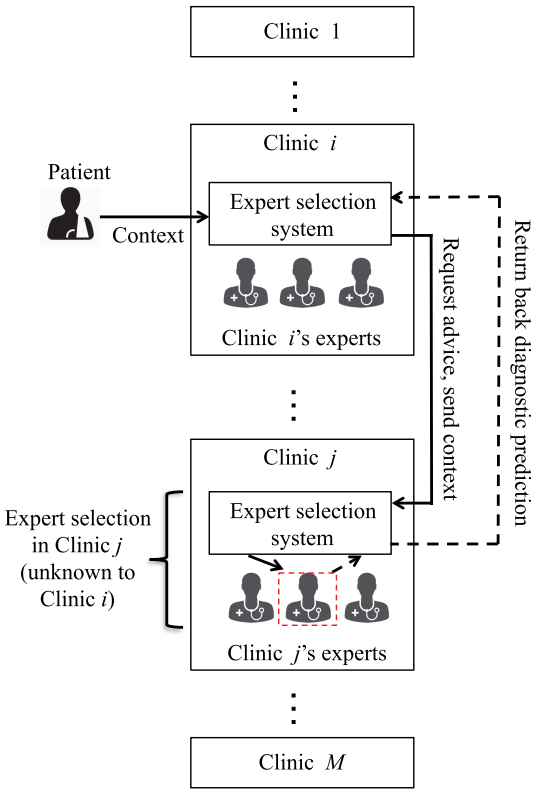

Fig. 3. Cooperative MABs for matching patients with experts.

features can be used as contexts to personalize the patient-expert matching. In such a setting, context-dependent quality of each expert must be learned through repeated interaction.

As an example, assume that there are M clinics, indexed by $i \in \mathcal{M} := \{1, 2, \ldots, M\}$, and E_i experts in clinic i indexed by $e \in \mathcal{E}_i := \{1, 2, \ldots, E_i\}$. Clinic i keeps an estimate of the quality/accuracy of its own experts $\hat{\pi}_e(\boldsymbol{x})$, $e \in \mathcal{E}_i$ and the quality/accuracy of the other clinics $\hat{\pi}_j(\boldsymbol{x})$, $j \in \mathcal{M} - \{i\}$ for each patient context $\boldsymbol{x} \in \mathcal{X}$. Here, $\hat{\pi}_j(\boldsymbol{x})$ is an estimate of the quality/accuracy of the best expert of clinic j given $\boldsymbol{x}$. The 3-phase learning structure proposed in Ref. 27 allows clinic i to form accurate estimates of this quantity without requiring it to know from which expert of clinic j the predictions come from.

Since learning and keeping separate quality estimates for each $\boldsymbol{x} \in \mathcal{X}$ is inefficient, clinic i partitions $\mathcal{X}$ into finitely many sets $\{p_1, \ldots, p_J\}$ of identical size. This partition is denoted by $\mathcal{P}$. Quality estimates are kept and updated for each $p \in \mathcal{P}$. Hence, for any context $x \in p$, $\hat{\pi}_e(\boldsymbol{x}) = \hat{\pi}_e(p)$ and $\hat{\pi}_j(\boldsymbol{x}) = \hat{\pi}_j(p)$. The granularity of $\mathcal{P}$ determines the learning speed and accuracy. The estimates converge quickly when the number of elements in $\mathcal{P}$ is small. On the other hand, the approximation error of $\hat{\pi}_e(\boldsymbol{x})$ and $\hat{\pi}_j(\boldsymbol{x})$ is higher when $\mathcal{P}$ is coarse, since the estimation is done using contexts more dissimilar to $\boldsymbol{x}$. The optimal way to set $\mathcal{P}$ is discussed in Ref. 27.

When a new patient arrives to clinic i, the learning algorithm first retrieves the context of the patient, which is denoted by $x_i(t)$. Then, it computes the set in the partition $\mathcal{P}$

that $x_i(t)$ belongs to, which is denoted by $p_i(t)$. Based on the accuracy of the estimates $\hat{\pi}_e(p_i(t))$, $e \in \mathcal{E}_i$ and $\hat{\pi}_j(p_i(t))$, $j \in \mathcal{M} - \{i\}$, clinic i will choose the clinic or expert to train, explore or exploit. The rule of thumb is to exploit only when the clinic is sufficiently confident about the quality of its own experts and the other clinics for the new patient. As seen in Fig. 3, this operation does not require clinic i to know the experts of the other clinics. All requests made by clinic i will be handled by the learning algorithm responsible for the expert selection system at clinic j. This allows the clinics to function autonomously, while also helping each other utilize their own resources when necessary.

In another application of cooperative contextual bandits, a network of multimedia content aggregators is considered.[11] These aggregators, each serving different types of users and having access to different types of multimedia content, learned to cooperate with each other in order to match their users with the right content.

4.2. *Ensemble of MABs*

In another strand of literature (See Refs. 28 and 29) ensemble learning methods are introduced to learn from multiple data sources. Data is processed locally by *local learners* (LLs) and the prediction of the LLs are sent to an *ensemble learner* (EL) which produces the final prediction. While majority of the related works treat LLs as black-box algorithms, in Ref. 3 a joint learning approach for LLs and the EL is developed. In this method, LLs implement a contextual MAB algorithm to learn their best classifier. On the other hand, the EL uses the dynamic meta-data provided about the LLs in order to combine their predictions.

An important application of this work is medical diagnosis,[30] in which predictions about the health condition of a patient are made by a set of distributed learners (or experts) that have access to different parts of the patient data. For instance, consider the situation described in Fig. 4, in which experts make local predictions based on different chunks of the patient's multi-modal medical data, which includes the *electronic health record* (EHR), mobile application data, lab test results and screening test results. This data is processed by multiple experts (some of which can be computerized diagnostic decision support systems) with different skills and different access rights to the data. First, each expert makes its own local diagnostic prediction. Then, these predictions are combined by an EL to produce a final prediction. Finally, after the true diagnosis (label) is revealed both the EL and the LLs update their prediction strategy.

The goal in this application is to maximize the expected number of correct predictions (over all patients arrived so far) by designing a decentralized learning algorithm, by which both the LLs and the EL update their prediction strategy on-the-fly. For instance, such an algorithm, which achieves the optimal learning rate is proposed in Ref. 3. The main idea behind this algorithm is to simultaneously enable LLs to learn

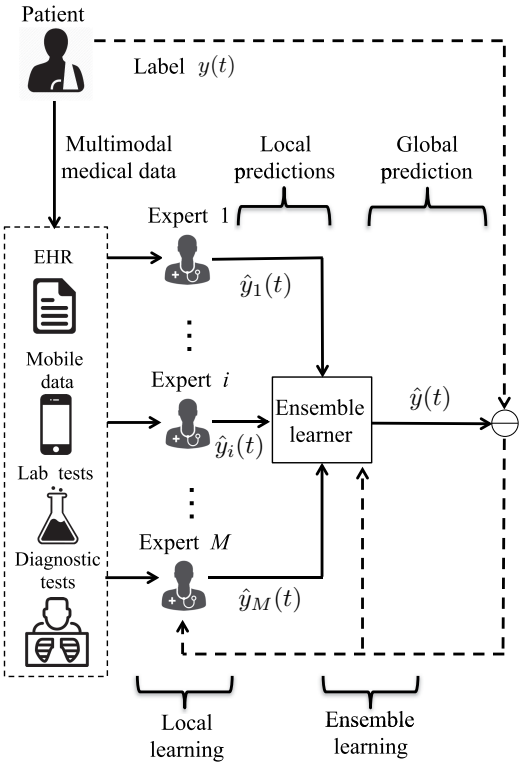

Fig. 4. Ensemble of MABs for medical diagnosis.

their best context-dependent prediction strategy by applying a contextual MAB learning algorithm and enable the EL to optimally fuse the predictions of the LLs by using a parameter-free ensemble learning algorithm.

It is shown in Ref. 3 that such a learning algorithm achieves prediction accuracy that is much higher than the prediction accuracy of the individual LLs or the prediction accuracy of an EL that works together with LLs that do not learn over time.

Apart from medical diagnosis, this method can also be used in big data stream mining. The benefit comes from both computation and memory savings as well as guaranteed performance bounds. Basically, it is shown in Ref. 3 that the memory requirement of the LLs in the proposed approach only increases sublinearly in the size of the dataset. On the other hand, the memory requirement for the EL is constant, since it only requires to learn the weights of the LLs.

5. Staged MABs

Apart from the task of selecting the right classifiers to maximize the prediction accuracy, in numerous semantic computing applications such as personalized education and clinical decision support, each decision step involves selecting multiple actions (e.g., selecting various materials to display or various tests to be performed) whose reward is only revealed after the entire action sequence is completed and a

decision is made to stop the action sequence and (possibly) take a final action (e.g., take a final exam, perform a surgery or finalize a treatment decision). For instance, in personalized online education, a sequence of materials can be used to teach or remind students the key concepts of a course subject and the final exam is used as a benchmark to evaluate the overall effectiveness of the given sequence of teaching materials. In addition, a sequence of intermediate feedbacks like quiz and homework grades can be used to guide the teaching examples online. These feedbacks can be combined into a context. For instance, a student's context can include fixed features such as her CGPA, as well as dynamic features such as the percentage of questions about a subject that she answered correctly. Most of the prior works on contextual bandits do not take into account such dynamic features.

Since the order of actions taken in the above problem affects the reward, previously discussed methods are incapable of addressing this problem. Such learning problems can be efficiently solved using staged MAB methods.[12,13] In a staged MAB, the learning algorithm selects the next actions to take in each round based on the estimated marginal gain of that action, while also taking into account the estimation error due to finitely many past observations. It is shown that this type of greedy action selection results in sublinear regret with respect to an approximately optimal benchmark, when the reward function is adaptive submodular.[31]

6. MABs, Regret Bounds and Confidence Intervals

One major advantage of applying MAB methods for semantic computing applications is their strong theoretical performance guarantees both in terms of regret and confidence intervals. Upper bounds on the regrets (defined in the introduction section) of various MAB methods are given in Table 1. While all methods achieve sublinear in T regret, the convergence speed of the time averaged regret values differ for these methods. Specifically, when adaptive contexts method is used, the time order of the regret depends only on the number of relevant context dimensions d_{rel}, which is in general much smaller than d. While the regret bounds characterize the long-run performance of MAB methods, confidence intervals provide information about the trustworthiness of the selected classifier. Basically, the confidence interval of classifier f at time slot t is a range of values $[L_f(t), U_f(t)] \subset [0,1]$ such that $\pi_f(\boldsymbol{x}(t))$ lies within $[L_f(t), U_f(t)]$ with a very high probability. Here, $L_f(t)$ is called the *lower confidence bound* and $U_f(t)$ is called the *upper confidence bound*. Numerous works provided data-size dependent confidence intervals and confidence bounds for MAB methods.[3,11,32,33] As expected, the confidence interval shrinks as the number of past data instances increase, depending on the risk awareness of the semantic computing application, more confident classifiers can be preferred over classifiers that perform better on average.

Table 1. Upper bounds on the regrets of various MAB methods.

Nonadaptive contexts Ref. 3	$O\left(T^{\frac{\gamma+d}{2\gamma+d}}\log T\right)$
Adaptive contexts Ref. 8 ($\gamma = 1$)	$O\left(T^{\frac{2+2d_{\text{rel}}+\sqrt{4d_{\text{rel}}^2+16d_{\text{rel}}+12}}{4+2d_{\text{rel}}+\sqrt{4d_{\text{rel}}^2+16d_{\text{rel}}+12}}}\log T\right)$
Cooperative contextual MAB Ref. 27	$O\left(T^{\frac{2\gamma+d}{3\gamma+d}}\log T\right)$

References

[1] D. Simons, Consumer electronics opportunities in remote and home healthcare, *Philips Res.* (2008).

[2] C. Tekin and M. van der Schaar, Active learning in context-driven stream mining with an application to image mining, *IEEE Trans. Image Process.* **24**, 3666 (2015).

[3] C. Tekin, J. Yoon and M. van der Schaar, Adaptive ensemble learning with confidence bounds, to appear in *IEEE Trans. Signal Process* (2016).

[4] C. Tekin, O. Atan and M. van der Schaar, Discover the expert: Context-adaptive expert selection for medical diagnosis, *IEEE Trans. Emerg. Top. Comput.* **3**, 220 (2015).

[5] J. Yoon, C. Davtyan and M. van der Schaar, Discovery and clinical decision support for personalized healthcare, to appear in *IEEE J. Biomed. and Health Inform* (2016).

[6] Y. Cao and Y. Li, An intelligent fuzzy-based recommendation system for consumer electronic products, *Expert Syst. Appl.* **33**, 230 (2007).

[7] L. Song, C. Tekin and M. van der Schaar, Online learning in large-scale contextual recommender systems, *IEEE Trans. Serv. Comput.* **9**, 433 (2016).

[8] C. Tekin and M. van der Schaar, RELEAF: An algorithm for learning and exploiting relevance, *IEEE J. Sel. Top. Signal Process.* **9**, 716 (2015).

[9] C. Tekin, S. Zhang and M. van der Schaar, Distributed online learning in social recommender systems, *IEEE J. Sel. Top. Signal Process.* **8**, 638 (2014).

[10] A. M. Eskicioglu and E. J. Delp, An overview of multimedia content protection in consumer electronics devices, *Signal Process: Image Commun.* **16**, 681 (2001).

[11] C. Tekin and M. van der Schaar, Contextual online learning for multimedia content aggregation, *IEEE Trans. Multimedia* **17**, 549 (2015).

[12] J. Xu, T. Xiang and M. van der Schaar, Personalized course sequence recommendations, *IEEE Trans. Signal Process.* **64**, 5340 (2016).

[13] C. Tekin, J. Braun and M. van der Schaar, eTutor: Online learning for personalized education, *IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)* (2015), pp. 5545–5549.

[14] G. Mateos, J. A. Bazerque and G. B. Giannakis, Distributed sparse linear regression, *IEEE Trans. Signal Process.* **58**, 5262 (2010).

[15] T. Lai and H. Robbins, Asymptotically efficient adaptive allocation rules, *Adv. Appl. Math.* **6**, 4 (1985).

[16] P. Auer, N. Cesa-Bianchi and P. Fischer, Finite-time analysis of the multiarmed bandit problem, *Mach. Learn.* **47**, 235 (2002).

[17] K. Liu and Q. Zhao, Distributed learning in multi-armed bandit with multiple players, *IEEE Trans. Signal Process.* **58**, 5667 (2010).

[18]T. Lu, D. Pál and M. Pál, Contextual multi-armed bandits, *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)* (2010), pp. 485–492.
[19]L. Li, W. Chu, J. Langford and R. E. Schapire, A contextual-bandit approach to personalized news article recommendation, *Proc. 19th Int. Conf. World Wide Web* (2010), pp. 661–670.
[20]C. J. Watkins and P. Dayan, Q-learning, *Mach. Learn.* **8**, 279 (1992).
[21]W. A. Gardner, Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique, *Signal Process.* **6**, 113 (1984).
[22]A. Slivkins, Contextual bandits with similarity information, *J. Mach. Learn. Res.* **15**, 2533 (2014).
[23]J. Langford and T. Zhang, The epoch-greedy algorithm for contextual multi-armed bandits, *Adv. Neural Inf. Process. Syst.* **20**, 1096 (2007).
[24]I. Zliobaite, Learning under Concept Drift: An Overview, Technical Report, Vilnius University, Faculty of Mathematics and Informatics (2010).
[25]J. Gao, W. Fan and J. Han, On appropriate assumptions to mine data streams: Analysis and practice, *Proc. IEEE ICDM* (2007), pp. 143–152.
[26]A. Garivier and E. Moulines, On upper-confidence bound policies for non-stationary bandit problems, arXiv:0805.3415.
[27]C. Tekin and M. van der Schaar, Distributed online learning via cooperative contextual bandits, *IEEE Trans. Signal Process.* **63**, 3700 (2015).
[28]N. Littlestone and M. K. Warmuth, The weighted majority algorithm, *30th Annual Symp. Foundations of Computer Science* (1989), pp. 256–261.
[29]Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *Computational Learning Theory* (1995), pp. 23–37.
[30]C. Tekin, J. Yoon and M. van der Schaar, Adaptive ensemble learning with confidence bounds for personalized diagnosis, *AAAI Workshop on Expanding the Boundaries of Health Informatics using AI (HIAI '16)* (2016).
[31]V. Gabillon, B. Kveton, Z. Wen, B. Eriksson and S. Muthukrishnan, Adaptive submodular maximization in bandit setting, *Adv. Neural Inf. Process. Syst.* 2697 (2013).
[32]Y. Abbasi-Yadkori, D. Pál and C. Szepesvári, Improved algorithms for linear stochastic bandits, *Adv. Neural Inf. Process. Syst.* 2312 (2011).
[33]D. Russo and B. Van Roy, Learning to optimize via posterior sampling, *Math. Oper. Res.* **39**, 1221 (2014).