This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2936565, IEEE Transactions on Knowledge and Data Engineering

A Privacy-Preserving Distributed Contextual Federated Online Learning Framework with Big Data Support in Social Recommender Systems

Pan Zhou, *Member, IEEE*, Kehao Wang, *Member, IEEE*, Linke Guo, *Member, IEEE*, Shimin Gong, *Member, IEEE*, Bolong Zheng, *Member, IEEE*

Abstract—Nowadays the booming demand of big data analytics and the constraints of computational ability and network bandwidth have made it difficult for a stand-alone agent/service provider to provide suitable information for every user from the large volume online data within the limited time. To handle this challenge, a recommender system (RS) can call in a group of agents to collaborate to learn users' preference and taste, which is known as a distributed recommender system (DRS). DRSs can improve the accuracy of a traditional RS by requesting agents to share information with each other. However, it is challenging for DRSs to make personalized recommendations for each user due to the large amount of candidates. In addition, information sharing among agents raises a privacy concern. Thus, we propose a privacy-preserving DRS in this paper, and then model each service provider as a distributed online learner with context-awareness. Service providers collaborate to make personalized recommendations by learning users' preferences according to the user context and users' history behaviors.We adopt the *federated learning* framework to help train a high quality privacy- preserving centralized model over a large number of distributed agents which is probably unreliable with relatively slow network connections. To handle big data scenario, we build an item-cluster tree to deal with online and increasing datasets *from top to the bottom*. We further consider the structure of social network and present an efficient algorithm to avoid more performance loss adaptively. Theoretical proofs show that our proposed algorithm can achieve sublinear regret and differential privacy protection simultaneously for service providers and users. Numerical results confirm that our novel framework can handle increasing big datasets and strike a trade-off between privacy-preserving level and the prediction accuracy.

Index Terms—Recommender system, differential privacy, online learning, federated Learning, big data, distributed and scalable model, cloud computing, mobile edge computing.

1 Introduction

1.1 Motivation

A Recommender system (RS) can understand users' preferences and recommend desirable items to them. As users' tastes vary with patterns, humans tend to receive the items similar to those they have shown interests before as well as the ones that other similar behavioral person likes [1]. As a result, the massive increasing data in the form of remarks, ratings, reviews, ranks, complaints, opinions, claims, and features about items (product, device, event, and service) on the web can be used for making correct decision and recommendation. These information often contain historical feedbacks from distinct users and the related similarities between items and users. If published, it is extremely beneficial for big data analysts to design effective data mining and machine learning approaches. However, they are usually sensitive and private, the direct releasing of which might throw threats on the safety of users and service providers resided in the RS. To achieve a balance between the reasonable uses of data information and users' privacy, policymakers must apply some of the most fundamental concepts of privacy law [2], and this is specially important for the big data technologies upon the development of RSs.

During the big data era, data information is usually massive, heterogeneous and evolves over time from different sources. The term "Big Data" is usually shown by 4V dimensions: Volume, Variety, Velocity, and Veracity [3]. In a RS, volume is encoded by the massive amount of data used to generate recommendations. Variety means data are extracted from different sources (e.g., blogs, Facebook, Twitter). Velocity stands for the speed of data generated online. Due to data volume, it becomes gradually hard for RSs to find useful and suitable information for users, and so does the other three dimensions. Hence, a RS with big data support aims to assist users to make beneficial choices from lots of alternatives is highly desirable. In addition, due to the variety of big data, it is hard to satisfy the tastes of all kinds of individuals. Thus, a RS should be able to provide personalized recommendations to different types

- *The work is supported by NSFC under Grant 61972448, 61672395, 61911540481, NSF under grant ECCS-1710996 and IIS-1722731.*

- *Pan Zhou is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China. (Corresponding author: Pan Zhou)*
- *Kehao Wang is with the Hubei Key Laboratory of Broadband Wireless Communication and Sensor Networks, Wuhan University of Technology, Wuhan 430070, China.*
- *Linke Guo is with Department of Electrical and Computer Engineering, Clemson University, SC 29634, USA.*
- *Shimin Gong is with School of Intelligent Systems Engineering, Sun Yat-sen University, 510275, Guangzhou, China.*
- *Bolong Zheng is with School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. E-mails: panzhou@hust.edu.cn, kehao.wang@whut.edu.cn, linkeg@clemson.edu, gong0012@e.ntu.edu.sg, bolongzheng@hust.edu.cn.*

of individuals. Furthermore, considering the velocity of big data, the computational and bandwidth constraints caused by the high volume of data prevent a stand alone service provider accessible to all the useful information (e.g., the information possessed by its competitive service providers). Therefore, to promote total performances, service providers are more willing to obtain low-rate information shared by other service providers. We can establish an online social network of service providers (agents) to achieve this goal, since online social network can allow a large scale of different agents to cooperate and coordinate [4]. This kind of RSs are known as distributed recommender systems (DRSs) [5], where a group of agents cooperate to discover relevant information and predict user behavior. However, the study on DRSs is an emerging and immature field. To stimulate our ideas, we provide a compelling and forthcoming reality application as follows.

Mobile edge computing (MEC), as an emergent technology, allows the computing services to be conveyed to edge nodes in the mobile social networks controlled by mobile base stations [6]. The architecture of MEC constitutes geo-distributed and inter-connected edge nodes that equipped with servers and storage units. Mobile users will offload their heavy computation tasks into the edge nodes. Users mainly send service request encoded as *contexts* to edge nodes. Their frequent requests of social and media services call for caching functionality in the edge nodes. So, a DRS is inherently necessary. The edge nodes support uploading new social and media data (items) into its storage units. Privacy is also a core concern among edge nodes and mobile users. Plus, much information is shared among edge nodes controlled by a central trusted system which ensures the security protection of designed distributed system. To tackle all these issues, authors in [42] point out that *federated learning* can be used as an operating system for edge computing.

In summary, DRS is promising to improve the recommendations performance. However, as we can see in the related work section that seldom data mining and machine learning algorithms could fully support the 4V dimensions of big data. Furthermore, the privacy issue is now becoming a widely concern for distributed systems (e.g., [7], [8]). Hence, it is an urgent research topic for DRSs. On the one hand, agents can share information to get mutual benefits. An agent in the online social networks can be any individual or company who can provide a useful service. For example, in products selling systems (e.g., Amazon and Taobao), different companies can help each other to sell products. Agents can obtain much greater rewards through such cooperation. On the other hand, directly releasing such sensitive information raises both agents' and their users' privacy concern. Therefore, we are necessary to consider two different privacy leakages: users' background information (context) and agents' item repository. The challenge herein can be concluded as: how to build a DRS with big data support which can make personalized recommendations, and how to guarantee the utility of information without disclosing the privacy of individuals and agents.

To solve the aforementioned issues, we design a contextual distributed online and privacy-preserving big data processing framework based on the tree structure, in which most connected agents can incorporate to make personalized predictions on the information sharing network by deploying privacy-preserving mechanism. Then we let a set of agents connected mutually by a fixed network. And each agent experiences inflows of users into its server/platform. Once the agent receives the arrival of certain user, it will recommend an item from its own database to the user. Technically, our proposed social DRS is based on contextual multi-armed bandit (CMAB) [45]. And CMAB measures the level of uncertainty and estimated performances of each arm (or item) by setting a related variable, produced according to the historical records and updated at every round. Then, it chooses items with the highest reward at every round based on historical feedback of selected items.

1.2 Related Work and Technical Comparison

Our work belongs to *social recommender systems*. Generally speaking, RSs deploy either content filtering approach [15] or collaborative filtering approach [16]. The content filtering approach establishes a profile for each user and item. However, this kind of profiles might be difficult to collect [14], [15]. The collaborative filtering approach relies merely on past user behavior, which results in cold start [21] problems when there are few historical records [16], [17], [19].

Nowadays, the lack of big data analytic technologies has become another challenge for RS. Representive traditional RSs based on users' social profile and relationship are in [18], [22]–[24]. However, almost all of them do not support extremly large datasets in both users' and items' categories. Although a recent such solution is shown in [25], it does not support dynamic incoming online data and changing users' contexts. All these challenges make classic recommender approaches infeasible in many practical scenarios.

The *Multi-armed Bandit (MAB)* method, as a promising solution, can achieve the vital exploration-exploitation trade-off in RS to promote performances. A plethora of works have been done on MAB algorithms [26]–[29], [53]. Some of them have focused on DRS [27]–[29]. For instance, Buccapatnam *et al* [27] present a RS by utilizing an online learning approach with UCB1, in which agents can share with each other to improve the accuracy for predicting on *fixed-size* and small-scale datasets. The identificaiton of applying MAB in the big data analytics is originated from the work [11], where the proposed UCT algorithm is based on the UCB1 and Monte-Carlo Tree Search (MCTS) and it can cope with potentially infinite many of arms. Works [12], [13] are representative progresses. However, all of them are *not context-awareness* which limit their applications in RSs.

Lots of previous works [26], [28], [29] have discussed about CMAB. For example, Cem *et al* [28], [29] introduce a DRS to achieve predictions based on personal demands for with a fixed size of item set. However, the proposed context partition approach is static that will cause massive computational complexity if it works in the large-scale dataset. Therefore, for big data analysis, it is clearly not practical. So Song *et al* introduce a RS which can partition context adaptively [26] for a single agent. However, the item-cluster tree is built from *bottom to the top*, that completely restrains the number of items. Therefore, it can not handle big data analysis, too. Additionally, the one-agent RS [26] can not deal with big data from decentralized or multimedia resources.

In our problem, we adopt the basic idea of MCTS, which can be used to analyze the item subspace rather than a single item to solve the problem of ever-increasing (and approaching infinite) item space size. To handle big data scenario, we propose an item-cluster tree structure that can expand these item-clusters adaptively from top to bottom instead of from bottom to up over time [26]. Our *top-to-bottom* structure supports recommendation on large-scale and increasing datasets, since all agents make recommendations on an item-cluster level rather than a single item level [26]. As a result, the component in the tree is scalable for all the agents which achieves big data analysis. Additionally, due to the dynamically expanding item-cluster tree, our approaches can support increasing datasets in real applications, while others [18], [22]–[24], [26], [28], [29], [38] cannot. We can model the selection of a child node in the cover tree we built every round as an MAB problem independently. However, comparing to [11]–[13], we have devised an adaptive context space partition model in our CMAB model with *context-awareness*. It evolves over time to guarantee that agents can explore current arriving user's context based on corresponding data of previously arrived users with the most similar contexts. Thus, it provides up-to-date personalized item recommendations. Furthermore, we utilize a threshold technique (detailded in Section 3.1) from [48] in our CMAB model to obtain the best online learning regret performance to date. In addition, our algorithm is a distributed framework that explores the agent cooperation and network structure (detailded in Section 4) to greatly improve the learning performance in the DRSs.

When it comes to the *privacy-preserving* problems in DRSs, the application of anonymity is wide and useful [34]. But in fact, common anonymity algorithms may impact potential utility of big data in high dimensions [24], [32], [33]. Furthermore, because of anonymity, users will be re-confirmed once adversaries collude mutually or auxiliary information is accessable [31]. Additionally, some existing algorithms often utilize cryptography [37] to ensure privacy not attacked. Although these approaches can guarantee security, they often lead to high communication costs and computation complexity that affects big data analysis greatly. Differential privacy (DP), as a promising technology, can deal with the problems of privacy issues [9], has been utilized in RSs in several researches. It is not required in DP for the adversaries' background information. Then DP only obtains related query results given by the database manager are not sensitive to changes (deletions or insertions) of individual data entries. That is to say when users' published data satisfies DP, while other attackers request the database manager with some tools helping data analysis, and the request results will be nearly similar even the user's records are not stored locally. Therefore, the users' privacy cannot be inferred and DP when applied in *large-scale* datasets [10] has a slight influence on the prediction accuracy. As a result, applying DP with BDA methods can guarantee both privacy and prediction accuracy. In this paper, we adopt DP as our privacy-preserving strategy. In [38], DP is applied to recommend tasks in a social graph. But [38] considers the social connections between agents in the social network and private information of users' preference, and we fcous on the privacy of users' context and agents' item repository.

To obtain scalable and reliable distributed solutions, we adopt the *federated learning* framework [41] in our setting to train a centralized model efficiently when the training data is distributed on lots of agents each with relatively slow and unreliable network connections. Federated learning is a very recent and promising research direction in artificial intelligence research under the "General Data Protection Regulation" [43], see its detailed concepts and applications in [42]. It main idea is to keep a centralized model at cloud or trusted third party (TTP) (or curator), and the changes are summarized by each local agent as a small focused update to the curator using privacy-preserving communication scheme. At the same time, the curator immediately averages the changes with other agents' updates to promote the shared model. All the training data remains on the local agent, and no updates are stored individually in the curator.

In this paper, we apply DP to provide an overall privacy guarantee on the model being trained from user data located at each agent. Using the idea of federated learning, each agent will deploy a Laplace mechanism to train the historical records [35] locally. When a query for shared information from its one-hop neighbors arrives, the TTP forwards the request to them and keep the privacy by deploying an Exponential mechanism [36]. The TTP only keeps an overall privacy-preserving tree-based learning model, and the overall prediction on TTP is generated by combining the individual predictions of this 'ensemble' of local models. Hence, it is a scalable solution for large-scale distributed network scenarios. Different from traditional approaches by adding noise to each related record, which will lead to great distortions, we introduce an adaptive binary tree-based noise aggregation method to guarantee DP and avoid the performance loss simultaneously. Furthermore, agents provide online learning results based on users' preferences with an Exponential mechanism. Furthermore, our theistical proofs confirm that our method can ensure DP of users' sensitive personalized contexts and agents' shared information. Additionally, we take the impact of the network structure into account.

1.3 Contributions

Our contributions are as follows:

1) We propose a *privacy-preserving* federated online learning algorithm with context-awareness for *DRSs* which supports big data. Specifically, the "Veracity", "Variety" and "Velocity" (besides the basic "Volume" features) of the big data "4V" dimensions and the key issue of "privacy" in the social DRSs are well addressed for the first time as a whole solution in our proposed framework.

2) The DRS partitions users' context space adaptively to make personalized recommendations. Numerically experimental results demonstrate that our proposal can support real-world increasing big datasets and outperform other state-of-the-art privacy-preserving schemes and online learning methods.

3) By defining rigorous attack models and the Exponential and Laplace mechanisms into DRSs, our algorithms can converge to the optimal strategy and guarantee privacy protection over time.

4) We introduce federated learning in our privacy-preserving DRSs, which progressively updates only one

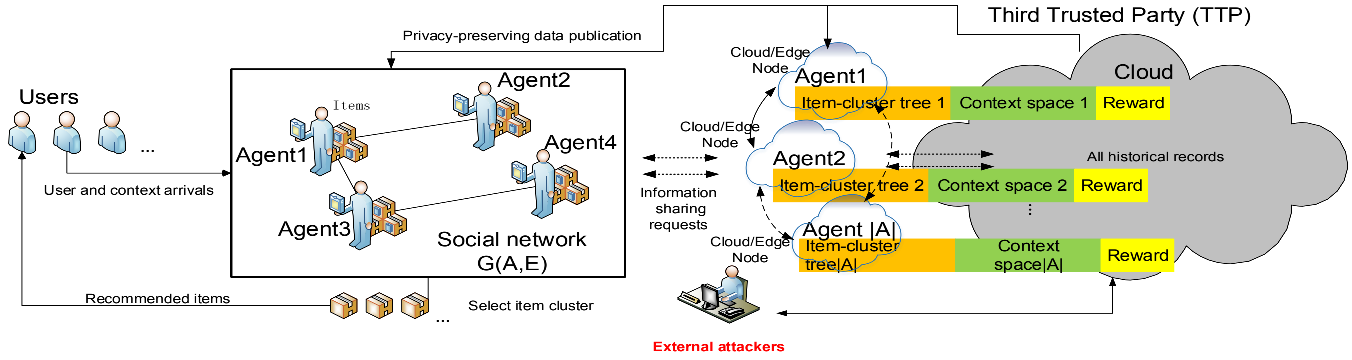


Fig. 1: System Model.

centralized model residing in the TTP. It provides smarter models, less power consumption and lower latency as well as ensuring privacy, which is suitable for unreliable and relatively slow network connections with superb scalability.

5) By exploring the network topology, it is an adaptive algorithm with the structure of online learning social network, which can accelerate learning speed and reduce the performance loss extensively.

The paper is organized in the following. Section 2 formalizes the problem. In section 3 and 4, we propose our algorithms and give our detailed theoretical analysis. In Section, simulation results are showed. Section 6 concludes the paper.

2 Problem Formulation

2.1 System Model

As shown in Fig. 1, our tree-based online learning privacy-preserving and contextual distributed framework consists of basic components as follows: *agents*, *items*, *users*, *trusted third party*.

Agents. We model the agents as distributed nodes in a graph firstly. Then we denote the set of agents as $W = \{1, 2, \cdots, |W|\}$, connected in a fixed network defined as a graph $G = (W, E)$. And E represents the set of edges. Then we denote all the connection between agents as a adjacency matrix $e(x, y)_{x,y \in W} \in \{0, 1\}$. If $e(x, y) = 1, e(x, y) \in E$, x is y's one-hop neighbor. In Fig. 2, we can observe $e(1, 2) = e(1, 3) = e(3, 4) = 1$. An agent can be a service provider in practice. For instance, a video service provider could offer more suitable videos to their users by sharing different video web applications.

Items. Each agent processes massive items in its repository. Each item is featured as a d_i-dimensional vector, and we denote every entry of the vector as one feature of a certain item. And the item feature vector is extracted from a d_i-dimension space I, then each dimension of it demonstrates one feature of the item.

Item-cluster Tree Structure. To handle big data scenario, we create a binary *item-cluster tree* structure $\mathcal{S}^{\mathcal{I}}$ for item space $\mathcal{I}$. It can diminish the scale of analyzed items every round for each agent in the networks. The item-cluster tree $\mathcal{S}^{\mathcal{I}}$ can expand infinitely and we denoted the i^{th} node at depth h as (h, i). Furthermore, we constrict the index i of nodes at depth h between 1 and 2^h. Obviously, $(h+1, 2i-1)$ and $(h+1, 2i+1)$ are the left child node and right child node of (h, i), respectively. Then in $\mathcal{S}^{\mathcal{I}}$, each node represents an item-cluster. For instance, as Fig. 2 shows, the root node $(0, 1)$ covers $\mathcal{I}$, but its right child $(1, 2)$ and left child $(1, 1)$ cover two different item subspaces of I measured by dissimilarity function D_i with same size. The depth of tree $\mathcal{S}^{\mathcal{I}}$ is denoted as $\mathcal{H} = \max_{(h,i) \in \mathcal{S}^{\mathcal{I}}} h$, and we define the area connected with cluster (h, i) as $A^{\mathcal{I}}_{h,i}$, which satisfies some conditions: $\forall h \geq 0, 1 \leq i, i' \leq 2^h, A^{\mathcal{I}}_{h,i} \cap A^{\mathcal{I}}_{h,i'} = \phi, A^{\mathcal{I}}_{h,i} = A^{\mathcal{I}}_{h+1,2i-1} \cup A^{I}_{h,2i+1}$.

Specially, our proposed item-cluster tree structure can recommend items based on the cluster level, which reduces the time complexity greatly. Compared to other methods which recommend items on a single item level, we can reduce the cost of obtaining appropriate items from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$ using our approach.

Users. Considering a time-slotted system, users arrive sequentially at each agent with his/her context. Each user's context in $\mathcal{C}$ is featured by the context vector c. Each user context vector is extracted from a d_c-dimensional context space $\mathcal{C}$, and every dimension of the context vector represents a related feature.

Adaptive context partition. Each subspace of $\mathcal{C}$ is modeled as a d_c-dimensions hypercube whose side length is k^{-m}, and m means the level of context subspace. Then the depth of context space is defined as $L = \max_{P \in \mathcal{C}} m_P$, where P is a subspace in C and l_P means the level of P. Because of dynamically increasing online dataset, we define a threshold for each context subspace to control its partition. When the number of context exceeds the threshold, we will partition this context space and the hypercube's each side is divided into k uniform parts (more details in next section) after the partition. Without loss of generality, we normalize $k = 2$ and $\mathcal{C} = [0, 1]^{d_c}$ for convenience.

Trusted Third Party (TTP). In order to prevent vulnerable behaviors impacting the DRS, we define the trusted third party (TTP) to handle it as shown in Fig.1. For example, the TTP in the MEC should be a selected trusted edge node (agent) or the main cloud in the hierarchical architecture, which would meet the latency requirements for mobile applications. We make training of the DP contextual tree-based model built at agents' servers to collaborate at the server site of TTP to build a global federated learning model. The training process of such a global federated learning model at each round usually contain the following four steps: 1) agents locally compute the contextual tree-based model and send the DP results to the server in the TTP; 2)

TTP performs secure aggregation of only models without learning information about any agent; 3)TTP send back the aggregated models to agents; 4) agents update their respective model with the DP results.

As noticed, the only responsibility of TTP is to provide the privacy-preserving federated learning model. In addition, when an agent requests information shared by its one-hop neighbors, the TTP will help add some Laplace noises on related information, then send the related average value which are conveyed back to current agent, but TTP does not store any data in its server. Tasks such as calculating the Φ-value in the accuracy prediction phase, recommending items in the item recommendation phase and training the local DP contextual tree-based model are accomplished in each agent's side. When an agent does not share information, the circle of executing the online algorithms is solely completed on itself. Furthermore, we assume that the TTP is trusted fully, so its privacy leakage is not included in our model.

Considering a system with the running rounds of our algorithm $n \in \{1, 2, \cdots\}$, a user with current context information arrives at each round. Let t_n represent the beginning time of round n, with the index of the agent, we can identify each item, user, and user's context information and reward uniquely under this absolute time. And at round n, we introduce $u^a(t_n)$ and $c^a(t_n)$ to represent the user and context information arriving at agent $a \in W$. Let $r^a(t_n)$ and $i^a(t_n)$ be the obtained reward and selected item at current round. In Fig. 1, the workflow of our system consists of four phases, and which are demonstrated from a single agent's perspective as follows.

i) *Strategy Adaptation:* Initially, our system determines whether or not an agent needs to tend for other agents for help based on its prior performance. If agent a has recommended an item that causes bad reputation (low empirical reward) for $u^a(t_{n-1})$ in the prior round, then agent a will request for information sharing with its one-hop neighbors through TTP. However, these attackers can easily infer the repositories according to the related information. Furthermore, each agent manages to protect its repository's privacy from other agents (competitors). This is because some items may bring large benefits but others only cause serious budget deficit. Therefore, the shared information is usually released utilizing a Laplace mechanism, which ensures the privacy of agents' repositories.

ii) *Performance Prediction:* In the accuracy prediction phase, at round n, a user $u^a(t_n)$ with context $c^a(t_n)$ arrives at the agent a; then, the agent learns the preference of $u^a(t_n)$ based on some historical data from the users whose context information is similar to user $u^a(t_n)$. It computes and learns the performances of each item cluster in its own item-cluster tree and predict each cluster's related performances according to Assumption 1 in subsection 2.3.

iii) *Item Recommendation:* In the item recommendation phase, agent a recommends an item $i^a(t_n)$ to user $u^a(t_n)$ based on the prediction in the accuracy prediction phase. However, as will be mentioned in next subsection, there are attackers which tend to expose user's sensitive context (e.g. gender, location, income level, etc.), by observing the recommended item to the user. An individual in the real world completely can be identified if these sensitive information are exposed. Thus, the agent uses an Exponential mechanism to recommend the suitable item instead of directly recommending, which may obtain the highest reward. Then, the TTP obtains a true reward $r^a(t_n)$ based on the behaviors/feedbacks of user $u^a(t_n)$. And the TTP provides a cloud storing all the records as shown in Fig. 1.

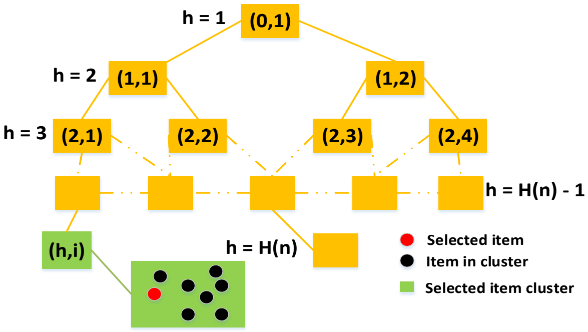


Fig. 2: Item-cluster Tree.

iv) *Item-cluster Tree Structure and Context Space Update:* Since the amount of historical data increases quickly over time and it is impossible to analyze all the historical data to do the predictions and recommendations, we need to restrict the size of analyzed component at each round. In addition, new data arrivals at each round (both side observations from one-hop neighbors and self-observations) need to be recorded. Thus, the item-cluster tree structure and context space are updated after current round. In the update phase, the context space becomes denser as time goes by and new leaf nodes is added to the item-cluster tree.

2.2 Adversary Model

There are two different types of adversaries: internal (i.e., people such as users and agents participating in this system) and external adversaries (i.e., people outside this system). In this paper, we consider one type of the internal attacks from agents and the external adversaries. The internal attacks are from the TTP (e.g., the TTP colludes with others to expose users' and agents' sensitive information) and users (e.g., vicious users ruin an agent's reputation by offering unreasonable rewards, and even invite others to give false high rewards for promoting an agent's reputation) are not included in this work. The attack models of two adversaries are described as follows.

2.2.1 Honest but Cunning Agents

The agents connected via social network $G(W, E)$ collaborate to make personalized recommendations of items to users by sharing information through G. The agents are totally honest to the TTP about their own information which can select suitable items based on proposed algorithm. And we assume that agents will ensure the privacy of any user's context information. Then, we discuss the condition without any privacy-preserving mechanism. At each round n, assuming that $i^a(t_n) \in (h_n^a, i_n^a)$ and $c^a(t_n) \in \mathcal{C}_n$ for any agent $a \in W$ which requests information sharing, agent a will send a query to the TTP complying the work flow of the proposed algorithm. This query will ask the TTP to let agent a know the number of data arrivals where the contextual arrivals are in the same context subspace $\mathcal{C}_n$ when the recommended items are in the same item cluster (h^a_n, i^a_n). In addition, the sum of the rewards based on these arrived data will be sent to agent a.

2.2.2 *Vicious Attackers*

Vicious attackers in the real world manage to recognize a user using some skills. They usually attack based on the previous records and utilize a data analysis program by sending similar requests disguised as a normal agent to obtain the rewards from this current user bypassing the TTP, then can infer the user's sensitive context information.

2.3 Performance analysis

In the subsection, we introduce some vital concepts for clarification.

Dissimilarity Function and Subspace Diameter: We define the dissimilarity function over item space $\mathcal{I}$ as a positive mapping D_i: $\mathcal{I}^2 \rightarrow A$ where $D_i(i, \vec{i}) \geq 0$ and $D_i(i,i) = 0$ for any $i, \vec{i} \in \mathcal{I}$. And the metric D_i in the Euclidean space $\mathcal{I}$ can be any norm. Furthermore, it is similar that we assume that the user context space $\mathcal{C}$ is provided with a function D_c to show dissimilarity such that $D_c(c, \vec{c}) \geq 0$ and $D_c(c,c) = 0$ for any $c, \vec{c} \in \mathcal{C}$. For $\forall i, \vec{i} \in \mathcal{I}$ and any subspace $R \in \mathcal{I}$, then the *diameter* of subspace is defined as $\mathcal{DIAM}^{\mathcal{I}}_R = \sup_{i,\vec{i} \in R} D_i(i, \vec{i})$ in subspace R.

Reward Production: At round n, for any agent $a \in W$, a reward $r^a(t_n)$ is obtained based on the current contexts and items from an unknown distribution. Feedbacks are given according to the users' behavior interacting with any agent a. For example, we consider the rewards from an online media content sharing system (e.g., Youtube), once someone clicks the content then selects it as one of the favorite contents, then $r^a(t_n) = 1$, $r^a(t_n) = 0$ instead. We assume that the reward is identically distributed with independence (i.i.d.), because random rewards can be drawn depending on former context information and items from an unknown distribution.

Suboptimal Cluster: Denoting the expected reward as $\mu_{i,c}$ for item i with context information c, the optimal reward for the user with context c as expected is defined as $\mu^*_c = \max_{i \in \mathcal{I}} \mu_{i,c}$, but it is acquired only in hindsight. The *suboptimal factor* of cluster (h,i) in the item-cluster cover tree is defined by $\Delta_{h,i}$ in the following: $\Delta_{h,i} = \max_{c \in \mathcal{C}} \mu^*_c - \max_{x \in (h,i)} \mu_{x,c}$. Once there is an item $i^* \in (h^*, i^*)$ and context $c \in \mathcal{C}$ stored in the system, and $\mu^*_c - \max_{x \in (h^*,i^*)} \mu_{x,c} \neq 0$, then we denote (h^*, i^*) as a *suboptimal node*. If $\mu^*_c - \max_{x \in (h^*,i^*)} \mu_{x,c} = 0$, it is an *optimal node*. As for our system with context-awareness, the reward of an item from users with similar context information is assumed to be nearly similar [40], [44]. Like in [26], [45], [47], we can formalize this assumption as the Lipschitz conditions as follows:

Assumption 1. *(Lipschitz conditions)* (a) (*Lipschitz condition for contexts*) For any context $c, \vec{c}$ and item $i \in \mathcal{I}$ respectively, it satisfies that $| \mu_{i,c} - \mu_{i,\vec{c}} | \leq L_c D_c(c, \vec{c})$, where L_c is a positive Lipschitz constant.

(b) (*Weak Lipschitz condition for items*) For $\forall$ item $i, \vec{i} \in \mathcal{I}$ and context $c \in \mathcal{C}$, we assume $| \mu_{i,c} - \mu_{\vec{i},c} | \leq \max\{D_i(i, \vec{i}), \mu^*_c - \mu_{\vec{i},c}\}$. And this assumption is fully reasonable in practical scenarios. For instance, assume that $d_c = 2$, then each context vector represents users' age and financial level. A 50-year-old user earning \$ 7500 per month and a 40-year-old user earning \$ 7800 per month may provide the similar low reward for luxuries. Though the realistic situations might not be always i.i.d., we can also bound a non-i.i.d. process [28] by utilizing two i.i.d. processes.

When we explore the item-cluster tree (more details in next section), the capability of each item node is usually limited. Therefore, we set an upper dissimilarity bound for all the items. Additionally, since in the item space $\mathcal{I}$ the items are discretely distributed, for all the items in each node, so there exists a relatively lower dissimilarity bound. This assumption is formalized in the following:

Assumption 2. For $\forall A^{\mathcal{I}}_{h,i} \in A^{\mathcal{I}}_{0,1}$, $\exists \psi_s > \psi_{s_1} > 0, \gamma_s \in (0,1)$, and the item-cluster's diameter at depth h is bounded as: $\psi_{s_1} \gamma^h_s \leq \mathcal{DIAM}^{\mathcal{I}}_{A_{h,i}} \leq \psi_s \gamma^h_s$.

Regret Computation on Performance Analysis: Let $\mu^a(t_n)$ be the expected reward of $i^a(t_n)$ without using Exponential mechanism and $\mu^a_E(t_n)$ is the reward of $i^a(t_n)$ as expected when Exponential mechanism is applied. Then we compute the one step *regret* for any $a \in W$ at time t_n, and we denote it as $\Delta_{t_n} = \mu^*_{c^a(t_n)} - r^a(t_n)$, to measure the performance loss for selecting the sub-optimal items. Then we denote the expectation of cumulative regret based on the network structure $G(W,E)$ as :

$$\mathbb{E}[R(n)] = \mathbb{E}[\sum_{a \in W} \sum_{i=1}^{n} |\Delta_{t_i}|] = \sum_{a \in W} \sum_{i=1}^{n} |\mu^*_{c^a(t_n)} - \mu^a_{E(t_i)}|,$$

where $\mu^*_{c^a(t_n)}$ is the optimal expected reward for user $u^a(t_n)$ with context $c^a(t_n)$. If our system can achieve sub-linear regret (i.e., $\mathbb{E}[R(n)] = O(n^\pi), \pi < 1$), we can infer that the strategy can converge to an optimal strategy (i.e., $\lim\limits_{n \to \infty} \frac{R(n)}{n} \to 0$). Furthermore, our main aim is to minimize the cumulative regret and protect privacy of users and agents from attacks in this paper.

3 Proposed Algorithm

3.1 Algortihm Description of T-PriDO

In this subsection, we propose our Tree-based Privacy Preserving Distributed Online Learning algorithm (T-PriDO). We define some important notations firstly, and then we represent the item cluster selected by agent a as (h^a_n, i^a_n). Next, we let $\mathcal{C}_n$ be the context subspace $c^a(t_n)$ at round n and denote l_n as the level of $\mathcal{C}_n$. Furthermore, we introduce the probability event: $\mathcal{K}^a_{h,i}(\tau)$ in Alg. 1 (Line 6), and $\mathbb{I}_{\mathcal{K}^a_{h,i}(\tau)}$ is an indicator function of event $\mathcal{K}^a_{h,i}(\tau)$. Then the depth of $\mathcal{T}^{\mathcal{I}}_a(n)$ which is the item-cluster tree of agent a at round n is denoted as $H(n)$, and we let $T^a_{h,i}(n) = \sum_{\tau < n} \mathbb{I}_{\mathcal{K}^a_{h,i}(\tau)}$. So we get $Q^a_{h,i}(n) = \sum_{b \in \mathrm{N}_a} T^b_{h,i}(n)$ (Line 6-7), where N_a is a set including agent a and its neighbors. We define the number of agents in N_a as $|\mathrm{N}_a|$, then denote the number of context arriving at certain context subspace $\mathcal{C}_n$ until round n as $N^a_{\mathcal{C}_n}(n)$ for agent a. Finally, we define $\hat{\mu}^a_{h,i}(n)$ as the empirical estimated mean reward of all items for agent a in cluster (h,i).

If a high estimated mean reward is received in a cluster, which means this cluster has a popular reputation and is

Algorithm 1: T-PriDO.

```
1  Input: network structure $G(W, E)$,
     $d_c, d_i, L_c, \psi_s, n, \gamma_s$;
2  Initialization: $t = 0, n = 1, \Phi^a_{1,1}(n) = \Phi^a_{1,2}(n) = infinity, \mathcal{T}^{\mathcal{I}}_a(n) = \{(0,1), (1,1), (1,2)\}, N^a_{\mathcal{C}_n}(n) = 0$;
3  Auxiliary procedure: TLM;
4  for $n = 1, 2, \cdots$ do
5      user $u^a(t_n)$ with context $c^a(t_n)$ at time $t_n$ arrives
         at agent $a$ and searches for the most relevant
         context subspace $\mathcal{C}_n$ ;
6      $\mathcal{K}^a_{h,i}(\tau) = \{(h^a_\tau, i^a_\tau) = (h,i), \tau < n, c^a(t_\tau) \in \mathcal{C}_n, a \in W, (h,i) \in \mathcal{T}^{\mathcal{I}}_a(n)\}$, and
         $Q^a_{h,i}(n) = \sum_{b \in \mathrm{N}_a} T^b_{h,i}(n) \sum_{\tau<n} \mathbb{I}_{\mathcal{K}^a_{h,i}(\tau)}$
7      if $\hat{\mu}_{h^a_n, i^a_n}(n-1) \geq NT$ then
8          $\hat{\mu}_{h^a_n, i^a_n}(n) = \frac{1}{Q_{h^a_n,i^a_n}(n)} \sum_{\tau \leq n} \mathbb{I}_{\mathcal{K}^a_{h,i}(\tau)} r^a(t_\tau)$;
9      else
10         $\hat{\mu}^a_{PC}(n) \leftarrow TLM(\mathcal{C}_n, a, (h_n, i_n))$;
11         $\hat{\mu}_{h^a_n, i^a_n}(n) = \frac{\hat{\mu}^a_{PC}(n)}{Q_{h^a_n,i^a_n}(n)}$;
12     for any leaf node $(h,i) \in \mathcal{T}^{\mathcal{I}}_a(n)$ do
13         Update $\Phi^a_{h,i}(n)$;
14     for any leaf node $(h,i) \in \mathcal{T}^{\mathcal{I}}_a(n)$ do
           /* $\mathbb{P}[(h^a_n, i^a_n) = (h,i)]$: the computed
              probability distribution using
              Exponential mechanism.                    */
15         $\mathbb{P}[(h^a_n, i^a_n) = (h,i)] = \frac{\exp \frac{\epsilon' \Phi^a_{h,i}(n)}{2\Delta\Phi}}{\sum_{(h,i) \in \mathcal{T}^{\mathcal{I}}_n} \exp\left(\frac{\epsilon' \Phi^a_{h,i}(n)}{2\Delta\Phi}\right)}$;
16     Select a leaf node $(h^a_n, i^a_n) \in \mathcal{T}^{\mathcal{I}}_a(n)$ based on
         computed probability distribution, then
         randomly recommend an item $i^a(t_n) \in (h^a_n, i^a_n)$
         for user $u^a(t_n)$ Thus, agent $a$ obtains the reward
         $r^a(t_n)$;
17     $N^a_{\mathcal{C}_n}(n) \leftarrow N^a_{\mathcal{C}_n}(n) + 1$;
       /* $\delta'_{l_n}$: the threshold of partition          */
18     if $N^a_{\mathcal{C}_n}(n) \geq \delta'_{l_n} = \lambda 2^{p l_n d_c} \ln n$ then
19         Partition space $\mathcal{C}_n$ into some subspaces;
20     if $Q_{h^a_n,i^a_n}(n) \geq \delta_{h^a_n} = \frac{\xi \ln n}{8\psi_s^2 \gamma_s^{2h^a_n}}$ then
21         $\Phi_{h^a_n+1, 2i^a_n-1}(n) = \Phi_{h^a_n+1, 2i^a_n}(n) = infinity$;
22         $\mathcal{T}^{\mathcal{I}}_a(n+1) =$
             $\mathcal{T}^{\mathcal{I}}_a(n) \cup \{(h^a_n+1, 2i^a_n-1), (h^a_n+1, 2i^a_n)\}$;
```

suitable for recommendations. However, if we just select items according to estimated mean rewards, we probably ignore other items that cannot offer a high reward currently but will have a good performance in the future. Therefore, we set a Φ-value for each item cluster to strike a trade-off between exploitation and exploration. The items in clusters are more likely to be chosen by agents with a bigger Φ-value (more details in Alg. 1). For $\forall(h,i) \in \mathcal{T}^{\mathcal{I}}_a(n)$, we denote the Φ-value of cluster (h,i) as $\Phi^a_{h,i}(n)$:

$$\Phi^a_{h,i}(n) = \begin{cases} \hat{\mu}^a_{h,i}(n) + \sqrt{\xi \frac{\ln n}{8Q^a_{h,i}(n)}} + \psi_s \gamma_s^h, & Q^a_{h,i}(n) > 0 \\ infinity, & \text{otherwise} \end{cases}$$

Hence, if there is an item cluster seldom selected, its Φ-value will gradually increase, so we can utilize its possible future better performance. When $\psi_s \gamma_s^h > \sqrt{\xi \frac{\ln n}{8Q^a_{h,i}(n)}}$ as the threshold condition, i.e., $Q_{h^a_n,i^a_n}(n) \geq \delta_{h^a_n} = \frac{\xi \ln n}{8\psi_s^2 \gamma_s^{2h}}$, it means that the cluster's size impacts more than the uncertainty from the randomness of the rewards on Φ-value. That is to say, the cluster (h,i) has been explored completely which represents that the number of data arriving in node (h,i) is large enough, so we should expand the item-cluster.

Now, we present the above mentioned workflow in subsection 3.1 phase by phase.

Phase 1: Strategy Adaptation. First, the system should determine if its strategy should be adopted (Line 7-11). A threshold NT is set by each agent to determine for shared information. The higher the NT, then more information will be shared. Due to the shared information (context, reward) may expose the repository of service providers, so we should use a differential private Laplace mechanism (Line 10) called *TLM* (described in subsection 4.4) to release the rewards obtained by each agent.

Phase 2: Performance Prediction. Then, T-PriDO needs to make predictions for all performances of each item cluster. And this is evaluated by Φ-values as aforementioned. To get Φ-values, T-PriDO first searches for the related contexts subspace $c^a(t_n)$ belongs to, which is denoted by $\mathcal{C}_n$ (Line 5). Then, in the item-cluster tree, all leaf nodes' Φ-values will be updated. Thus, the Φ-values reflect the related expected reward of items in current item-cluster.

Phase 3: Item Recommendation. Considering the selected item cluster might be shared with other agents or attacked, so it is unreasonable to select the item cluster with highest Φ-value that will expose user's sensitive context. Therefore, agent a, using Exponential mechanism (Line 14-15), chooses an item cluster according to the computed probability distribution. For any leaf node $(h,i) \in \mathcal{T}^I_a(n)$, $\mathbb{P}[(h^a_n, i^a_n) = (h,i)]$ (Line 15) is a score function, then Δ_Φ is the l_1 sensitivity of Φ-value. Due to Exponential mechanism, these attackers cannot infer the users' context accurately because the items might not be selected, although with the highest reward.

Phase 4: Item-cluster tree Structure and Context Space Update. The number of context arrivals in subspace $\mathcal{C}_n$ is updated over time in Alg. 1 (Line 17). When it in $\mathcal{C}_n$ exceeds the threshold δ'_{l_n} (Line 18-19), which means this current subspace $\mathcal{C}_n$ needs to be partitioned for limiting the related components' size. At last, $\mathcal{C}_n$ will be divided into these new built subspaces (e.g., in Fig. 3, $d_c = 2$, $\mathcal{C}_x$ at t_y is partitioned into four subspaces of level l_y). Finally, to obtain reward estimation of each item cluster accurately, we should set a threshold $\delta_{h^a_n}$ for agent a (Line 20-21) to judge whether (h^a_n, i^a_n) has been explored completely. Once $Q_{h^a_n,i^a_n}(n)$ exceeds $\delta_{h^a_n}$, which means item cluster (h^a_n, i^a_n) has been explored entirely and it will be partitioned uniformly then add new nodes to expand the tree (Line 21-22).

In short, our proposed item-cluster tree structure partitions the item space into more and more refined cluster level and the size of items in the child clusters levels will be smaller and smaller, which have similar rewards under the Lipschitz conditions. As time goes by, it finally seek out the last (and the deepest) leaf node that contains only one optimal item corresponding to a certain user context input.

3.2 Differential Private Framework.

The concepts about DP are available in [51]. Our proposed differential private framework consists of two privacy-

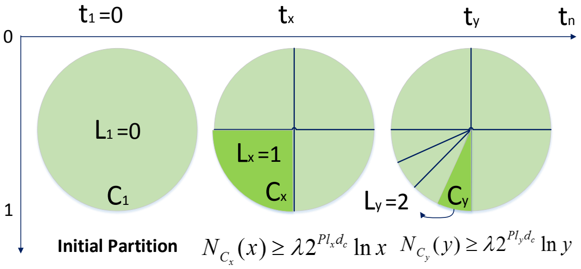

t1 =0
tx
ty
tn
0
1
L1 =0
C1
Lx =1
Cx
Ly =2
Cy
Initial Partition
$N_{C_x}(x) \ge \lambda 2^{Pl_x d_c} \ln x$
$N_{C_y}(y) \ge \lambda 2^{Pl_y d_c} \ln y$

Fig. 3: Context Space Partition.

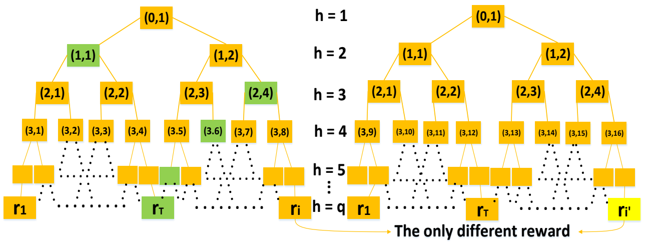


Fig. 4: Tree Based Approach of Adding Laplace Noise.

preserving modules: 1) the module for users and 2) the module for agents. The first module is achieved using an Exponential mechanism in Alg. 1 (Line 14-15). The second one is to protect the private information shared by all agents.

3.2.1 The Privacy-preserving Module for Users

When a cunning agent is curious about users' contexts, he/she just needs to send a query to the TTP composed of two pairs (i.e., C_n and (h_n^a, i_n^a)). For clarification, we present an Example 1 of attacks on users' contexts in [51]. Thus, we should apply Exponential mechanism to protect the best performing item cluster (Line 14-15). Theorem 1 below shows that we can guarantee DP of the context information from certain user.

Theorem 1. Algorithm 1 can guarantee ε-differential privacy for user's sensitive context information.

Proof: Detailed proof of Theorem 1 in [51]. □

Theorem 1 means that the query results are not sensitive to the changes of individual user's records, so users' sensitive context will not be inferred from the published results.

3.2.2 The Privacy-preserving Module for Agents

When agent a tends to its one-hop neighbors for shared information, if we do not apply any privacy-preserving mechanism, a neighboring agent can infer its item repositories after sending enough queries to the TTP. We present an Example 2 for attacks on agents' item repositories in [51].

Since requesting for shared information belongs to normal legitimate behavior and the level of NT is decided by the agent itself, the TTP cannot stop an agent to infer other agents' repositories. As a result, we should use a Laplace mechanism to prevent privacy of agents from being exposed. To ensure the privacy of information shared, we apply a naive approach in DP by adding noise to each reward. But this will cause much noise in the big data system, that causes shared information useless for other agents. Therefore, according to [49], [50], we present a Noise Aggregation algorithm by building a binary tree based on Laplace mechanism (TLM) in Alg. 2 in the next subsection.

3.3 Algorithm Description of TLM

At round n, while agent a asks for information sharing from its one-hop neighbors (Line 6-8), the binary tree $\mathcal{T}_{\mathcal{C}_n}^a(n)$ that stores the whole historical data information in the set of agents N_a in context subspace $\mathcal{C}_n$ till round n is built (Line 24). In Fig. 4, each leaf node contains a previous reward by an agent a in N_a, and all the related user's context in the subspace $\mathcal{C}_n$. And the sum of rewards rooted at current node in all the leaf nodes is stored in each internal node. Based on the absolute time of storing rewards, the leaf nodes are ranked from left to right. In Fig. 4, it shows that if $b, c \in \mathrm{N}_a$ such that $r_1 = r^b(t_i), r_T = r^c(t_j)$, where i and j are r_1's and r_T's local arriving round at agent b and c and $t_i < t_j$. Then, the subset $\Xi_{\mathcal{C}_n}^a(n)$ of disjoint nodes in $\mathcal{T}_{\mathcal{C}_n}^a(n)$, which cover almost all previous data till round n, will be built (Line 24). Then we choose one node at each depth of $\mathcal{T}_{\mathcal{C}_n}^a(n)$ (e.g., the green nodes) to obtain the subset $\Xi^a(n)_{\mathcal{C}_n}$ in Fig. 4. We let q be the depth of $\mathcal{T}_{\mathcal{C}_n}^a(n)$. To ensure its neighbors' repository safe (Line 7, 25-26), each sum of rewards stored in nodes in $\Xi_{\mathcal{C}_n}^a(n)$ are added Laplace noise, then we send these q sums to agent a with Laplace noise. For simplicity, we assume the number of data arriving at round n in subspace C_n as $T = 2^\nu \le n$, where $\nu \in \mathbb{Z}, \nu \ge 0$. We get $\mathcal{A}, \mathcal{A}'$ be two databases that differ in only one entry (i.e., $\|\mathcal{A} - \mathcal{A}'\| \le 1$). For $\forall \mathcal{C}_n$ and N_a, the binary tree denoted by $\mathcal{T}_\mathcal{A}$ based on $\mathcal{A}$ (e.g., tree in Fig. 4 on the left) and the binary tree denoted by $\mathcal{T}_{\mathcal{A}'}$ based on $\mathcal{A}'$ (e.g., tree in Fig. 4 on the right), have only one different leaf node (e.g., in Fig. 4, r_i and $r_{i'}$). And it is obvious that $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_{\mathcal{A}'}$ differ mostly in $\ln n$ reward sums.

```
Algorithm 2: TLM.
23 Input: $\mathcal{C}_n$, agent $a$, cluster $(h, i)$;
24 Initialization: $\mathcal{T}_{\mathcal{C}_n}^a(n) = \{(0,1)\}$;
25 Establish $\mathcal{T}_{\mathcal{C}_n}^a(n), \Xi_{\mathcal{C}_n}^a(n)$;
   /* $\hat{r}$: the sum of rewards of items stored
      in node $(x, y) \in \mathcal{T}_{\mathcal{C}_n}^a(n)$ and in cluster
      $(h, i)$;                                         */
26 $\hat{\mu}_{\mathcal{C}_n}^a(n) = \sum_{(x,y)\in\Xi_{\mathcal{C}_n}^a(n)} \left(Lap(\frac{\ln n|W|}{\varepsilon}) + \hat{r}\right)$;
27 Output: $\hat{\mu}_{\mathcal{C}_n}^a(n)$.
```

Furthermore, our TLM compared with traditional LMs, can obviously reduce Laplace noise from $\mathcal{O}(n)$ to $\mathcal{O}(logn)$. Therefore, it simultaneously guarantees to use the aggregated data and protect the privacy of agents. It demonstrates that T-PriDO can guarantee the privacy of agents' repositories in the following:

Theorem 2. Algorithm 1 can guarantee ε-differential privacy for all agents' revenue of rewards.

Proof: Detailed proof of Theorem 2 in [51]. □

Theorem 2 confirms that a service provider cannot draw anything about item repositories from the shared information, because the rewards from two distinct item clusters are

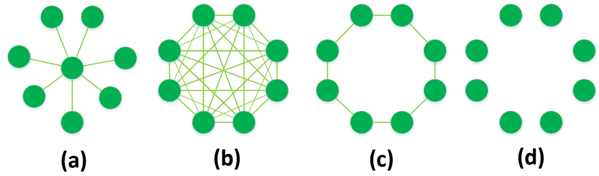


Fig. 5: (a) 8-agent star network; (b) 8-agent fully connected network; (c) 8-agent circular network; (d) 8 stand alone agents without connection.

highly similar. Thus, Theorem 1 and 2 confirm that T-PriDO can protect the privacy of service providers and users.

3.4 Regret Analysis

In this subsection, we analyze the prediction accuracy of Alg.1 . And the inaccuracy is from two aspects: 1) the noise we add which influences optimal selections from the agents, 2) performance losses on the early stage for exploring which leads to suboptimal predictions. We cannot avoid these inaccuracies, the expected cumulative regret bound is high in Alg. 1. Based on TLM, we bound the noise added to each Φ-value in the following lemma as $\mathcal{O}\left(\frac{(\ln n)^2}{\varepsilon}\ln(\frac{n\ln n}{\rho})\right)$ with the probability $\geq 1-\rho$.

Lemma 1. Given $G(W,E)$, $\forall a \in W$ and for $\forall(h,i) \in \mathcal{T}_a^{\mathcal{I}}(n)$, we denote the maximum of Laplace noise added to $\Phi^a_{h,i}(n)$ till round n as $\Upsilon_{h,i} = \frac{|W|(\ln n)^2}{\varepsilon}\ln(\frac{n\ln n}{\rho})$ with probability $\geq 1-\rho$.

Proof: Detailed proof of Lemma 1 in [51]. □

The lemma demonstrates that we can bound all added noise to each Φ-value owing to the tree-based noise aggregation mechanism. In addition, the lemma in the following means that if a suboptimal node only is chosen for several specific times, it can hardly be chosen by other agents. We denote the context subspace's smallest side length at round n as $D_{\mathcal{C}_n} = 2^{-L(n)}$, where $L(n)$ means the depth level of related context space.

Lemma 2. Given $G(W,E)$, for $\forall a \in W$ and any suboptimal node $(\vec{h},\vec{i})$, let $\varpi^a_{\vec{h},\vec{i}} = \min\{\tau \leq n : Q^a_{\vec{h},\vec{i}}(\tau) \geq \kappa_{h,i} = \lceil \frac{\xi\Upsilon_{\vec{h},\vec{i}}}{2(\Delta_{h,i}-\psi_s\gamma_s^h)^2}\}\rceil$, where $\Upsilon_{h,i}$ is the same notation as that in Lemma 1 and $\xi > 4$. Letting $\Delta_{(h^*,i^*)} = 0$, we have $\mathbb{P}\{\Phi^a_{h,i}(n) > \Phi^a_{h^*,i^*}(n), \forall n \geq \varpi^a_{h,i}\} \leq 2n^{-\frac{\xi}{4}}$.

Proof: Detailed proof of Lemma 2 in [51]. □

According to the above lemmas, we can finally bound the *upper cumulative regret* of T-PriDO as follows:

Theorem 3. Given $G(W,E)$, upper expected cumulative regret of Alg. 1 is

$$\begin{aligned}
&\mathbb{E}[R(n)] \leq 4nL_c\sqrt{d_c}\cdot n^{\frac{1}{d_c(1+p)}}(4\lambda\ln n)^{-\frac{1}{d_c(1+p)}}(\varrho+1) \\
&+(\frac{2\xi\ln n}{3\gamma_s\psi_s\psi_{s_1}^{d_0}}+\frac{64|W|G_u\xi\ln n}{3{\gamma_s}^2})(1+\frac{1}{\varrho})(\frac{\varrho L_c\sqrt{d_c}}{\psi_s})^{-d_0}\cdot \\
&(4\lambda\ln n)^{\frac{d_0}{d_c(1+p)}}\cdot n^{\frac{-d_0}{d_c(1+p)}}+ \qquad (1)\\
&\frac{24}{\varepsilon}\xi G_u|W|\frac{L_c\sqrt{d_c}}{\psi_s^2\psi_{s_1}^{d_0}}(\ln n)^{3+\frac{d_0+3}{(p+1)d_c}}n^{\frac{-d_0+3+2p}{p(p+1)d_c}}(\frac{\varrho L_c\sqrt{d_c}}{\psi_s})^{-d_s} \\
&+\frac{2|W|L_c}{\varepsilon}(\ln n)^3.
\end{aligned}$$

where $\xi > 6$, $\varrho = \frac{\psi_{s_1}\gamma^{\hat{H}}}{L_cD_{C_n}}$, and $\exists\eta \to 0$, we have

$$\begin{aligned}
\Delta &= (d_c-2)^2 - 4d_c(d_0-3) > 0, \\
p_1 &= \frac{-(d_c-2)+\sqrt{\Delta}}{2d_c} < \eta,
\end{aligned} \qquad (2)$$

and G_u is a parameter related to the network structure $G(W,E)$ among all the agents. $\kappa_{h,i}$ is the same notation compared to Lemma 2, and $G_u\kappa_{h,i}$ means the maximum number of reward samples at round n in item-cluster (h,i). Thus, the minimum number of samples each agent can obtain accessibly is $\kappa_{h,i}$ including the shared information from the neighbors (i.e., $Q^a_{h,i}(n) \geq \kappa_{h,i}$). In Fig. 5, from (a)-(d), we have $G_u = \lfloor\frac{|W|}{2}$ for star network, $G_u = |W|$ for stand alone agents without connection, $G_u = 1$ for fully connected network and $G_u = |W|-1$ for circular network. We denote the nearly optimal dimension of item space $\mathcal{I}$ as d_0 (details about Lemma 4 in [51]), and $d_0 = 0$ in most conditions [48].

Proof: Detailed proof of Theorem 3 in [51]. □

Remark 1. We can neglect the last term which is the regret by utilizing Exponential mechanism compared to the regret caused by inherent gap. Our proposal with Exponential mechanism cannot select an optimal action each round, but it selects the optimal action with the highest probability. And if p gets larger, then the first and second term increase, and the third added term will decrease. Because the first term keeps in the highest time order, with p increasing, the upper expected cumulative regret will also increase . This fully accords with our expectation and the current context space will be partitioned more slowly if p gets larger. Therefore, the overall similarities will decrease between arriving context $c^a(t_n)$ and contexts in $\mathcal{C}_n$ which may disturb agents to make personalized predictions accurately. However, if p decreases to a certain small degree, we cannot ensure the third term sublinear anymore. This is caused that when p is too small, it will make the context space be partitioned quickly, so there are only small amounts of related context information in $\mathcal{C}_n$. This will cause agents to learn historical records with a deficiency, which increases unreliable predictions. Additionally, there is a trade-off between prediction accuracy and the privacy-preserving level (i.e., smaller ε may lead to bigger upper cumulative regret). And when the connectivity in $G(W,E)$ (i.e., G_u decreases) is improved, the upper expected cumulative regret will decrease. The main reason is that more historical records are shared then there will be more sufficient analysis obtained if the network's connectivity becomes better.

Time and Space Complexity. We can divide the computational cost of T-PriDO into following parts: i) finding the context subspace C_n; ii) refresh Φ-values and the item-cluster structure; and iii) select a leaf node using Exponential mechanism. For the first part, the computational cost is decided by the level of C_n. And, we can know from the Lemma 3 in [51] that the level of C_n is restricted to $\mathcal{O}(\log_2 n)$. For the second part, T-PriDO needs to traverse all the historical records in C_n. Considering the worst case when all the historical results are distributed in C_n, the computational cost is $\mathcal{O}(n)$. In the third part, the computational cost depends on the number of leaf nodes which is bounded by $2^{H(n)}$. Then from the Lemma 8 in [51], we can know that the

Algorithm 3: DT-PriDO.

```
Input: $D_G, G(W, E)$;
Initialization: $t = 0, n = 1, H(n) = 0; \mathcal{T}_a^{\mathcal{I}}(n) = \{(0,1),(1,1),(1,2)\}; \Phi_{1,1}^a(n) = \Phi_{1,2}^a(n) = infinity$;
Further partition $G(W, E)$ based on $D_G$;
if agent $a \in D_G$ then
    Similar to Alg. 1 (Line 2-22);
else
    Follow the dominator's selections in its group;
```

depth is the item-cluster tree for any agent is restricted to $\mathcal{O}(\ln n)$. Thus, the computational cost for the third part is $\mathcal{O}(2^{\ln n})$. Updating the model from each agent, hence the computational cost up to round n in the federated learning is $\mathcal{O}(n \log_2 n + n^2 + n2^{\ln n})$. Since T-PriDO needs to store all the historical results locally at each agent, so the whole space complexity is $\mathcal{O}(n)$.

4 Proposed Adaptive Algorithm

In the network some agents are usually named as *dominators*, which are centrally located among other agents. The dominators usually have more one-hop neighbors compared to ordinary agents. Thus, we can get more accurate predictions from dominators, since dominators can observe more comprehensively with more neighbors. In this section, we design an adaptive Dominator centered Tree-based Privacy Preserving Distributed Online Learning Algorithm (DT-PriDO), that regards the dominators as the major controller but other agents mainly offer the side information in the network. Furthermore, the *dominator subset* $D_G \in W$ which consists of dominators in network $G(W, E)$. In this case, for $\forall a \in W \setminus D_G$, agent a will have more than one one-hop neighbor in subset D_G.

DT-PriDO works as follows: There is a dominator subset $D_G \in W$, then the network $G(W, E)$ is partitioned into $|D_G|$ groups, and for each group, it includes one dominator from the dominator subset D_G and its all one-hop neighbors. Finally, in the dominator subset D_G the whole agents entirely follow the same strategy of T-PriDO, but other agents follow their own dominators of the groups they belong to.

4.1 Privacy and Regret Bound Analysis

We obtain privacy results of Alg. 1 in the following.

Theorem 4. DT-PriDO can make sure ε-differential privacy for all the users' sensitive context information and service providers' item repositories.

Proof: DT-PriDO keeps independent of the network structure, which is similar in Theorem 1 and 2, then we obtain Theorem 4. □

Theorem 4 confirms that DT-PriDO can guarantee ε-privacy for both service providers and users similar to T-PriDO. To get upper expected cumulative regret of DT-PriDO, we can bound the expected number of selections for suboptimal clusters similar to Lemma 2.

Lemma 3. Given the network structure $G(W, E)$, for any suboptimal cluster (h, i) and dominator subset D_G, we get

$$\mathbb{E}[\sum_{a \in W} T_{h,i}^a(n)] \leq \frac{D_G \xi |W| (\ln n)^3}{2\varepsilon(\Delta_{h,i} - \psi_s \gamma_s^h)^2} + 1 + |W| + \frac{4|W|}{\xi - 4}, \quad (3)$$

where $|D_G|$ means the number of groups in $G(W, E)$ and $\xi > 4$.

Proof: Detailed proof of Lemma 3 of [51]. □

And Lemma 3 limits the expected number of samples that the agent can access. According to Lemma 3, we can obtain the upper cumulative regret bound of DT-PriDO.

Theorem 5. Based on the network structure $G(W, E)$ and a dominator subset D_G, we can get upper expected cumulative regret of DT-PriDO:

$$\begin{aligned}
&\mathbb{E}[R(n)] \leq 4nL_c\sqrt{d_c} \cdot n^{\frac{1}{d_c(1+p)}} (4\lambda \ln n)^{-\frac{1}{d_c(1+p)}} (\varrho + 1) \\
&+ (\frac{2\xi \ln n}{3\gamma_s \psi_s \psi_{s1}^{d_0}} + \frac{64|W|D_G \xi \ln n}{3{\gamma_s}^2})(1 + \frac{1}{\varrho})(\frac{\varrho L_c \sqrt{d_c}}{\psi_s})^{-d_0} \cdot \\
&(4\lambda \ln n)^{\frac{d_0}{d_c(1+p)}} \cdot n^{\frac{-d_0}{d_c(1+p)}} + \qquad (4) \\
&\frac{24}{\varepsilon} \xi G_u |W| \frac{L_c\sqrt{d_c}}{\psi_s^2 \psi_{s1}^{d_0}} (\ln n)^{3 + \frac{d_0+3}{(p+1)d_c}} n^{\frac{-d_0+3+2p}{p(p+1)d_c}} (\frac{\varrho L_c \sqrt{d_c}}{\psi_s})^{-d_s} \\
&+ \frac{2|W|L_c}{\varepsilon} (\ln n)^3,
\end{aligned}$$

where p, ϱ and c are constrained similar to Theorem 3.

Proof: Detailed proof of Theorem 5 in [51]. □

Remark 2. Based on Theorem 3 and 5, it is evident that DT-PriDO decreases the second term of upper expected cumulative regret to $\frac{|D_G|}{G_u}$ that of T-PriDO. Above conditions are caused by choosing the dominators as the reliable central makers in the network, which can provide more accurate predictions. When the connectivity of the network becomes worse, such as G_u increases, the impact produced will be obvious. Thus, DT-PriDO can improve the performance of the network though the connectivity is relatively low.

5 Numeral Results

5.1 Dataset Description

We utilize an dataset YFCC100M [52] in the real world, provided by Yahoo in 2014. There are more than 100 million media objects in YFCC100M composed of 100.2 million images and 0.9 million videos. And each entry contains some basic information including location, owner name, camera, title, media source, and tags. We utilize 34,8743 photos and 1821 videos annotated with detected concepts visually and some information of cameras. Then we let $d_i = 5$ to represent the dimension of item features including timestamp, media type marker (video =0, photo=1), longitude, latitude, and camera maker. Then the top 30 of 1,580 detected concepts and cameras in the dataset are utilized in our simulation. And by using Flickr API, we can obtain the information of user contexts and user behaviors (e.g., location, time, userID). For example, when a user adds a video to his/her favorite list or shares it with others, the reward will equal to 1. And we calculate the total rewards based on the browsing time. Let $t^a(n)$ be the longitude of time user $u^a(n)$ cost browsing $i^a(n)$ and $\hat{t}$ be the average time $u^a(n)$ cost browsing recommendations. Thus, if $t^a(n) \leq \hat{t}$, $r^a(n) = 0$, otherwise $r^a(n) = 1$. Then we set $d_c = 9$ as the dimensions of context features including timestamp, longitude, latitude, ID, age,

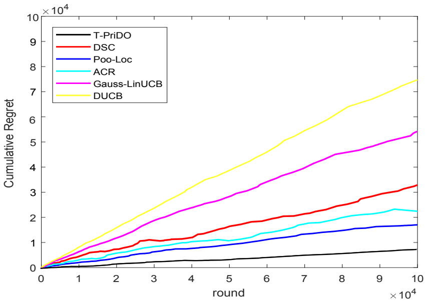


Fig. 6: Cumulative Regret-Static.

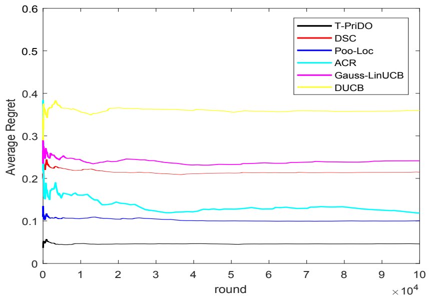


Fig. 7: Average Regret-Static.

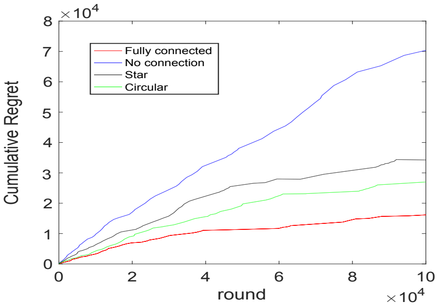


Fig. 8: Various Networks-Cumulative.

TABLE 1: Average Reward.

Algorithm	round $\times 10^4$							Gain[1]
	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	
Random	0.25	0.27	0.25	0.26	0.26	0.25	0.26	238%
UCB1	0.28	0.30	0.31	0.30	0.28	0.28	0.28	214%
DUCB	0.32	0.31	0.32	0.33	0.33	0.33	0.33	167%
LinUCB	0.45	0.46	0.47	0.47	0.48	0.48	0.48	83%
Gauss-LinUCB	0.46	0.49	0.48	0.48	0.49	0.50	0.51	73%
DSC	0.50	0.51	0.54	0.56	0.57	0.58	0.59	49.2%
ACR	0.56	0.54	0.57	0.57	0.59	0.65	0.62	41.9%
Poo-Loc	0.56	0.56	0.58	0.60	0.61	0.63	0.65	35.4%
T-PriDO (one agent)	0.58	0.62	0.63	0.65	0.67	0.69	0.69	27.5%
T-PriDO	0.57	0.60	0.61	0.70	0.78	0.85	0.88	

[1] The gain of T-PriDO over other algorithms in average reward.

salary, etc. We perform all experiments on the computing platform of one of the authors' university computing center. SSD cache of it is 1.24 TB and the CPU reach 18.37 TFlops. .

5.2 Comparison With Online Learning Algorithms

We compare our algorithms firstly with other related works to demonstrate the good performance of our proposed algorithm. First, we compare our proposed T-PriDO with those algorithms without context-awareness to show the importance of user's context for promoting system performance. The results are as follows: 1) Random: This algorithm chooses an item at each round randomly which is regarded as a benchmark for other related algorithms. 2) UCB1 [30]: This traditional MAB algorithm performs well with known fixed item size in advance to recommend the best items, which does not considers uses' personalization.

To demonstrate the importance of sharing, then we compare T-PriDO with two distributed online learning systems as follows: 3) DUCB: As a distributed online learning system, it uses UCB1 [27] without context information where agents share their observations bypassing a social network. 4) Poo-Loc: It is an MCTS distributed online learning system using POO in [13] without context information that only requires local smoothness with respect to the chosen partitioning. Furthermore, it is the state-of-the-art and the most powerful context-free MCTS algorithm, and we implement the same DP schemes in our setting to make fair comparison. In contrast, our T-PriDO also only requires local smoothness, but we introduced the threshold techniques (details in Section 3.1) to obtain better performance.

To confirm our algorithm can outperform other context-aware recommendation systems, we compare T-PriDO with three context-aware systems as follows: 5) DSC [28]: As a distributed online learning system, the context space in it is partitioned previously, and it keeps static over time. 6) ACR [26]: This is a context-aware centralized online learning system with some fixed item clusters. We utilize $K = 90$ item clusters in ACR for simulation. 7) LinUCB [53]: It is assumed that in contexts the rewards of items are linear, and this method is based on CMAB algorithm that is widely used in practical news artical recommendations, which utilizes the maximum index to recommend the arm. And we implement all these algorithms under our noising injecting and DP framework. 8) Gauss-LinUCB [54]: We also implement the latest DP version of LinUCB, named as "Gauss-LinUCB", which could apply either Gaussian noise or Wishart noise to achieve joint-differentially private algorithms and bound the regrets. We choose the Gaussian noise version in comparison. Intuitively, Gauss-LinUCB is a state-of-the-art and more advanced algorithm when compared with the LinUCB with the default DP framework in our setting, which should have better learning performance under the ε-differential privacy notion.

We evaluate the performance by computing the average reward till round n. It can decrease average reward by using a privacy-preserving mechanism or an increasing dataset. However, information sharing among agents can increase average reward. Therefore, for fair comparisons, we do not utilize the PPMs of T-PriDO in the comparison experiments and guarantee a static dataset. We create a 8-agent star network with one center for DSC, T-PriDO, and DUCB. Furthermore, all results are performed when $NT = 0.5$ except that shown in Fig. 14. And we also show the average reward of T-PriDO with one agent for fair comparisons. The comparison results are shown in TABLE 1.

Comparison results with context-free algorithms: As TABLE 1 shows, T-PriDO outperforms other algorithms without context-awareness. The results demonstrate that T-PriDO obtains a 214% performance gain on average reward, over UCB1, and it even achieves a 35.4% performance gain over the state-of-the-art context-free Poo-Loc algorithm. Notice that UCB1 and Poo-Loc learn faster than T-PriDO (converges faster), because the evaluation of the average reward on context-free algorithm is acquired from the historical records of various users, but a context-aware algorithm obtains the average reward according to users' certain context information. And it is obvious that context-aware algorithms can predict more accurate items with the number of users' arrivals increasing to a specific high degree. And we can observe that the context-free algorithms can converge faster than the algorithms with context-awareness before round $n = 4 \times 10^4$. However, the context-aware algorithms

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2936565, IEEE Transactions on Knowledge and Data Engineering

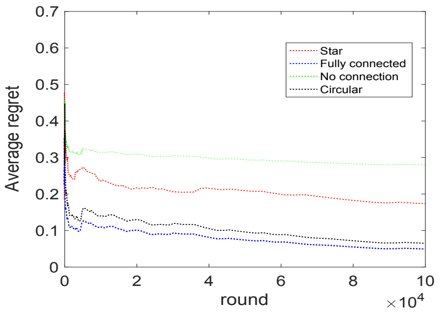


Fig. 9: Various Networks-Average.

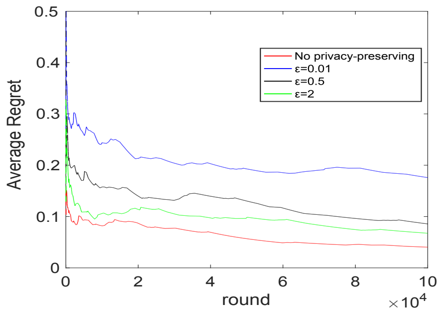


Fig. 10: Average regret-ε level.

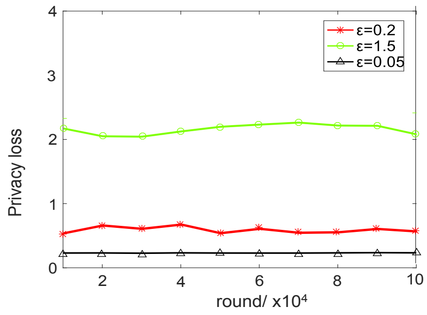


Fig. 11: Privacy Loss.

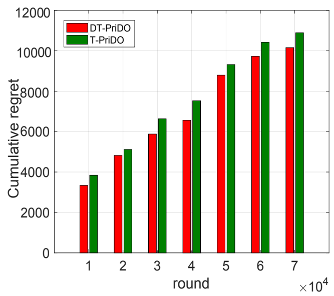


Fig. 12: Cumulative Regret.

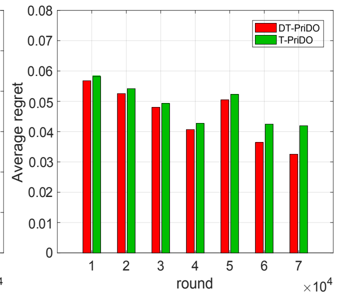


Fig. 13: Average Regret.

TABLE 2: Variance of Average Regret.

Algorithm	**round**$\times 10^4$									
	n=1	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
ACR	0.0021	0.0023	0.0022	0.0021	0.0021	0.0024	0.0025	0.0029	0.0032	0.0035
DUCB	0.0037	0.0041	0.0038	0.0035	0.0037	0.0037	0.0040	0.0045	0.0052	0.0055
DSC	0.0011	0.0013	0.0009	0.0010	0.0010	0.0011	0.0011	0.0009	0.0010	0.0009
T-PriDO	0.0010	0.0011	0.0011	0.0010	0.0009	0.0011	0.0011	0.0013	0.0015	0.0017
Gauss-LinUCB	0.0010	0.0009	0.0008	0.0007	0.0008	0.0008	0.0008	0.0007	0.0008	0.0007

can obtain a higher average reward than other context-free algorithms in a long term.

Comparison results with context-aware algorithms: As the results show, T-PriDO outperforms completely other context-aware algorithms. The average reward of T-PriDO up to $n = 7 \times 10^4$ is 83% over LinUCB, 73% over Gauss-LinUCB, 49.2% over DSC and 41.9% over ACR. Because T-PriDO aggregates information in the context space adaptively, but DSC always considers static fixed context subspace that impacts its online learning performance. As for LinUCB, the payoff function leads to inaccurate estimation of linear reward dissimilar to our proposal. In Gauss-LinUCB, the situation is similar to LinUCB, but Gauss-LinUCB has better learning performance than that of LinUCB. Note that DSC has a better learning efficiency that T-PriDO, but it learns less accurately. And DSC supports information sharing in online social networks, but it still obtain lower average reward than that of ACR with shared information. Themain reason is that T-PriDO and ACR both utilize a dynamic context partition approach contrary to the static one from DSC. Therefore, an adaptive method for context partition can ensure more accurate and reasonable predictions, although it cannot guarantee an optimal learning speed.

Furthermore, we provide the comparison results on cumulative regret and average regret with three most related algorithms in Fig. 6 and 7. And the results are similar to TABLE 1 as we expect. For instance, we can see that T-PriDO decreases in average regret by 24.83%, 27.93%, 47.45%, 61.17% and 76.67% when $n = 2 \times 10^4$ compared to PooLoc, ACR, DSC, Gauss-LinUCB, and DUCB, respectively. We can also obtain similar results of cumulative regret in Fig. 6. Additionally, we observe that teh learning speed of context-free algorithms is faster than that of context-aware algorithms, according to the slopes of the curves. The main example is DSC and the result also accords with the results in TABLE 1. In addition, we put the variance of average reward associated with DSC, DUCB, ACR and T-PriDO in TABLE 1. From TABLE 2, we can find that ACR and DUCB have higher variance than that of T-PriDO, DSC and Gauss-LinUCB. The results might result from the randomness of rewards. ACR and T-PriDO have adaptive context partition methods, but DSC utilizes a static one. Thus, it might be possible that as the degree of personalization increases, variance increases. As noticed, Gauss-LinUCB has the least variance of average regret. This is because the Gaussian noise based joint-differentially private algorithms, as a latest progress in DP, have less noise variance than that of the classic Laplace noise setting.

5.3 Evaluation Other Performance

Impact of Network Structure: To obtain the influence of network structure, we run our algorithm based on the static dataset with a network composed of 40 agents which is fully connected, a star network or a circular network. In Fig. 8 and 9 that T-PriDO will get smaller average regret and cumulative regret if connectivity in the network increases. This condition accords with our theoretical analysis, because G_l and G_u will decrease and each agent can obtain more useful information as the connectivity of network is improved.

Impact of Increasing Dataset: To investigate T-PriDO's ability to deal with the increasing dataset, we run DT-PriDO and T-PriDO on 40-agent star network without any privacy-preserving mechanism. Thus, we create an increasing dataset by selecting 2,000 videos and 200,000 images as the initial dataset, and at round $n = 4 \times 10^4$, add the rest filtered 80,000 images and 800 videos to the database. As shown in Fig. 13, the average regret of our proposal increases at round $n = 5 \times 10^4$ but it begins to decrease quickly after round $n = 6 \times 10^5$. We can infer from this that our proposal can support the increasing dataset adaptively. As shown in Fig. 12, from $n = 1 \sim 7 \times 10^4$, T-PriDO has a regret gain from 16% to 9% over DT-PriDO on the cumulative regret. It means that DT-PriDO can recommend

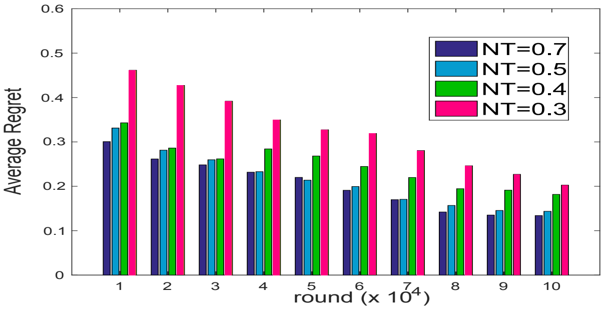


Fig. 14: Average Regret-NT levels.

items more accurately than T-PriDO in a star network, but the differences between them decrease gradually.

Impact of Privacy-Preserving Mechanism: Using the static dataset, at different privacy-preserving levels (different ε values), we measure the average accuracy of our proposals to demonstrate the impact on prediction accuracy with different privacy-preserving levels in Fig. 10 and TABLE 3. As shown in Fig. 10 and TABLE 3, we can observe that it achieves a balance between the privacy-preserving level and average accuracy. As the privacy-preserving level increases (ε decreases), each reward sum will be added more noise, which makes the shared information unreliable. While the privacy-preserving level exceeds a threshold, the regret will be so large that it will spend more time on converging to the optimal strategy, which means the design of privacy-preserving level is essential. Finally, to measure the privacy-preserving level, we utilize privacy loss $PL = \max_O \ln(\frac{\mathbb{P}[\mathcal{M}(D)=O]}{\mathbb{P}[\mathcal{M}(D')=O]})$ [9] to evaluate the privacy-preserving level of our methods. Based on Definition 1, $PL \leq \varepsilon$. And it is shown in Fig. 11 under the same experimental setting of T-PriDO in TABLE 2 that PL of T-PriDO keeps a low level, which accords with our theoretical results. Therefore, our approach has a high privacy-preserving level.

Impact of NT level: NT shows the frequency of communications among agents. and the higher NT is, more frequent the communications are. Thus, we evaluate the average regret at different values of NT in Fig. 14. Then we perform the experiments under a 40-agent star network. And we can see from Fig. 14 that average regret is proportional to the value of NT inversely, which infers that the prediction accuracy can be promoted based on distributed system.

6 Conclusion

We propose a novel distributed federated online learning algorithm with context-awareness and big data support for social DRS and design a framework to improve prediction performance adaptively. Then, we give theoretical analysis in details. The experimental results demonstrate that our proposed algorithms can achieve these following goals: suitable item recommendations, accurate predictions of users' preference, differential privacy protection for users' context, and differential privacy protection for agents' repository.

TABLE 3: Average Accuracy.

Algorithm	ε	round $\times 10^4$						
		$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
T-PriDO	0.01	41.92%	44.89%	47.72%	54.37%	55.91%	58.43%	60.26%
	0.5	70.13%	71.58%	72.11%	73.12%	75.83%	84.02%	86.21%
	2	90.32%	92.87%	93.96%	95.23%	96.13%	97.51%	98.12%
DT-PriDO	0.01	41.96%	42.45%	47.45%	53.21%	58.51%	60.42%	61.19%
	0.5	71.61%	77.32%	73.56%	76.65%	78.50%	86.21%	87.93%
	2	94.67%	96.05%	97.85%	97.99%	98.01%	98.82%	99.13%

References

[1] J. P. Verma, B. Patel, and A. Patel, "Big data analysis: recommendation system with Hadoop framework", *IEEE Proc. of CICT*, pp. 92-97, 2015.

[2] O. Tene, J. Polonetsky, "Big data for all: Privacy and user control in the age of analytics", *Nw. J. Tech. & Int. Prop.*, pp. 11, 2012.

[3] A. Gandomi, and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics", *IJIM*, vol. 35, no. 2, pp. 137-144, 2015.

[4] K.-C. Chen, M. Chiang, and H. Poor,"From technological networks to social networks," *IEEE JSAC*, vol. 31, no. 9, pp. 548–572, 2013.

[5] P. Giannikopoulos and C. Vassilakis," A distributed recommender system architecture," *IJWET*, vol. 7, no. 3, pp. 203-227, 2012.

[6] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie,"Mobile edge computing: A survey," *IEEE IOT Journal*, vol. 5, no. 1, pp.450-465, 2018.

[7] C. Li, P. Zhou, L. Xiong, Q. Wang, T. Wang, "Differentially Private Distributed Online Learning," *IEEE TKDE*, Vol.99, no.1, 2018.

[8] Q.Wang, M. Du, X. Chen, Y. Chen, P. Zhou, X. Chen, X. Huang, "Privacy- Preserving Collaborative Model Learning: The Case of Word Vector Training," *IEEE TKDE*, Vol.99, no.1, 2018.

[9] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *TCC*, vol. 9, no. 3-4, pp. 211-407, 2013.

[10] E. Klarreich, "Privacy by the Numbers: A New Approach to Safeguarding Data," *Quanta Magazine*, pp.10, 2012.

[11] L. Kocsis, C. Szepesvári, "Bandit based monte-carlo planning. In European conference on machine learning," Springer, Berlin, Heidelberg, pp. 282-293, 2006.

[12] M. Valko, A. Carpentier, R. Munos, "Stochastic simultaneous optimistic optimization," In Proc. of ICML, pp. 19-27, 2013.

[13] X.Shang, E. Kaufmann, M. Valko, "Adaptive black-box optimization got easier: HCT only needs local smoothness," In European Workshop on Reinforcement Learning, 2017.

[14] L. Sharma, and A. Gera, "A survey of recommendation system: Research challenges", *IJETT*, vol. 4, no. 5, pp. 1989-1992, 2013.

[15] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends", *Recommender systems handbook*, pp. 73-105, 2011.

[16] M. D. Ekstrand, J. T.Riedl, J. A Konstan, and others, "Collaborative filtering recommender systems", *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81-173, 2011.

[17] Y. Dou, H. Yang, and X.Deng, "A survey of collaborative filtering algorithms for social recommender systems.", *in Proc.of In Semantics, Knowledge and Grids (SKGD)*, pp. 40-46, 2016.

[18] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks", *in Proc. 4th ACM Conf. Recom. Syst.*, pp.135-142, 2010.

[19] C. Chen, X. Zheng, Y. Wang, F. Hong, and Z. Lin, "Context-aware collaborative topic regression with social matrix factorization for recommender systems", *in Proc. of Recomm. Syst.*, pp.135-142, 2010.

[20] G. Guo, J. Zhang, and N. Yorke-Smith, "TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings", *in Proc.of 29th AAAI*, pp.123-129, 2015.

[21] C. C. Chen, Y.-H. Wan, M.-C. Chung, and Y.-C. Sun, "An effective recommendation method for cold start new users using trust and distrust networks", *Inf. Sci.*, vol. 224, pp.19-36, 2013.

[22] S. Yan, K.J. Lin, X. Zheng, W. Zhang, and X. Feng, "An Approach for Building Efficient and Accurate Social Recommender Systems Using Individual Relationship Networks", *IEEE TKDE*, vol. 29, no. 10, pp. 2086-2099, 2017.

[23] C.-C. Hsu and M. Y. Yeh, "A General Framework for Implicit and Explicit Social Recommendation", *IEEE TKDE*, vol. 30, no. 12, pp. 2228-2241, 2018.

[24] D. Yang, B. Qu, P. Cudre-Mauroux, " Privacy-Preserving Social Media Data Publishing for Personalized Ranking-Based Recommendation", *IEEE TKDE*, vol. 31, no. 3, pp.507-520, 2018.

[25] D. Lian, Y. Ge, F. Zhang, N. J. Yuan, X. Xie, T. Zhou, and Y.X. Rui, "Scalable Content-Aware Collaborative Filtering for Location Recommendation", *IEEE TKDE*, vol. 30, no. 6, pp. 1122-1135, 2018.

[26] L. Song, C. Tekin, and M. van der Schaar, "Online learning in large-scale contextual recommender systems," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 433–445, 2016.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2936565, IEEE Transactions on Knowledge and Data Engineering

[27] S. Buccapatnam, A. Eryilmaz and N. B.Shroff, "Multi-armed bandits in the presence of side observations in social networks", *IEEE CDC,* 2013, pp. 7309–7314.

[28] C. Tekin and M. Schaar, "Distributed online big data classification using context information," *IEEE CCC,* pp. 1435-1442, 2013.

[29] C. Tekin, Z. Shaoting and M. van der Schaar, "Distributed online learning in social recommender systems," *JSTSP,* vol. 8, no. 4, pp. 638-652, 2014.

[30] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[31] A. Narayanan and V. Shmatikov, "How to break anonymity of the Netflix prize dataset" CoRR, 2006. [Online]. Available: http://arxiv.org/abs/cs/0610105

[32] J. Brickell and V. Shmatikov, "The cost of privacy: Destruction of data-mining utility in anonymized data publishing," *in Proc. 14th ACM SIGKDD KDD*, pp. 70-78, 2008.

[33] C. C. Aggarwal and C. Charu, "On k-anonymity and the curse of dimensionality," *in Proc. 31st VLDB*, 2005, pp. 901-909.

[34] S. Shang, Y. Hui, and P. Hui, "Beyond personalization and anonymity: Towards a group-based recommender system," *in Proc. 29th Annu. ACM Symp. Appl. Comput.*, 2014, pp. 266-273.

[35] C. Dwork, F. McSherry, K. Nissim, and A. Smith. "Calibrating noise to sensitivity in private data analysis," *Theory of Cryptography Conference*, pp. 265-284, 2006.

[36] F. McSherry and K. Talwar, "Mechanism design via differential privacy," *FOCS,* pp. 94-103, 2007.

[37] Z. Erkin and T. Veugen, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE TIFS*, vol. 7, no. 3, pp. 1053-1066, Jun. 2012.
A. Slivkins, "Contextual bandits with similarity information", *in Proc. of 24th ACL*, pp. 679-701, 2011.

[38] Z. Jorgensen and T. Yu, "A privacy-preserving framework for personalized, social recommendations," *in Proc. of EDBT*, pp. 571-582, 2014.

[39] N. Cesa-Bianchi and G. Lugosi, "Prediction, learning, and games," Cambridge, Cambridge University Press, 2006.

[40] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," *in Recom. Sys. Handbook, Springer US*, pp. 217-253, 2011.

[41] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv preprint arXiv:1610.05492, 2016.

[42] Q. Yang, Y. Liu, T. Chen, Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM TIST*, vol.10, no.2, pp.1-19, 2019.

[43] C. Tikkinen-Piri, A. Rohunen, J. Markkula, "EU General Data Protection Regulation: Changes and implications for personal data collecting companies," *Comp. Law & Sec.y Rev.*, vol.34, no.1, pp.134-153, 2018.

[44] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM TOIS,* vol. 23, no. 1, pp. 103-145, 2005.

[45] T. Lu, D. Pal, and M. Pal,"Contextual multi-armed bandits," *in Proc. of AISTATS*, pp. 485-492, 2010.

[46] R. Kleinberg, A. Slivkins, and E. Upfal,"Multi-armed bandits in metric spaces," *in Proc. of STOC*, pp. 681-690, 2008.

[47] A. Slivkins, "Contextual bandits with similarity informatio", *in Proc. of ACLT*, pp. 679-701, 2011.

[48] M. G. Azar, A. Lazaric, and E. Brunskill, "Online stochastic optimization under correlated bandit feedback," *in Proc. of ICML*, pp. 1557–1565, 2014.

[49] C. Dwork, M. Naor, and G. N. Rothblum, "Differential privacy under continual observation," *in Proc. of STOC,* pp. 715-724, 2010.

[50] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," *TISSEC*, vol.4, no.3, pp 1-23, 2011.

[51] P. Zhou, et al, [Supplementary], "A Privacy-Preserving Distributed Contextual Federated Online Learning Framework with Big Data Support in Social Recommender Systems", [Online]. Available: https://www.dropbox.com/s/2cnv1j4y050xxm7/TKDE19_Supp.pdf?dl=0

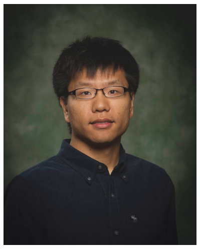
[52] B. Thomee, D. A. Shamma, G. Friedland, et al. "The new data and new challenges in multimedia research," *arXiv preprint arXiv:1503.01817,* vol. 1, no. 8, 2015.

[53] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextualbandit approach to personalized news article recommendation," *in Proc. of the 19th WWW*, pp. 661-670, 2010.

[54] R. Shariff, O. Sheffet,"Differentially Private Contextual Linear Bandits," *In Proc. of Neur-IPS*, pp. 4301-4311, 2018.

Pan Zhou(S'07–M'14) is currently an associate professor with School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the *Advanced Class* of HUST in 2006. He was a senior technical member at Oracle Inc, America during 2011 to 2013, Boston, MA, USA. His current research interest include: network security, machine learning and big data analytics, information networks.

Kehao Wang received the BS degree in Electrical Engineering, MS degree in Communication and Information System from Wuhan University of Technology, Wuhan, China, in 2003 and 2006, respectively, and Ph.D in the Department of Computer Science, the University of Paris-Sud XI, Orsay, France, in 2012. From Feb. 2013 to Aug. 2013, he was a postdoc with the HongKong Polytechnic University. In 2013, he joined the School of Information Engineering at the Wuhan University of Technology, where he is currently an associate professor. His research interests include: stochastic optimization, operation research, scheduling, wireless network communications, and embedded operating system.



Linke Guo (M14) received the BE degree in electronic information science and technology from the Beijing University of Posts and Telecommunications in 2008. He received the MS and PhD degrees in electrical and computer engineering from the University of Florida in 2011 and 2014, respectively. Since August 2014 to May 2019, he has been an assistant professor in the Department of Electrical and Computer Engineering, Binghamton University, State University of New York. He is currently an assistant professor in Department of Electrical and Computer Engineering, Clemson University. He is the co-recipient of Best Paper Award of Globecom 2015. His research interests include network security and privacy, social networks, and applied cryptography.

Shimin Gong (M'15) received the B.E. and M.E. degrees in Electronics and Information Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2008 and 2012, respectively, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, in 2014. He is currently an Associate Professor with the School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou, China. Before that, He was an associated researcher with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He has been the Lead Guest Editor for IEEE Transactions on Cognitive Communications and Networking – special issue on Deep Reinforcement Learning on Future Wireless Communication Networks. He is a recipient of the Best Paper Award on MAC and Cross-layer Design in IEEE WCNC 2019. His research interests include wireless powered IoT, deep reinforcement learning, backscatter communications and networking.

Bolong Zheng (M'14) is an Associate Professor in Huazhong University of Science and Technology (HUST). He received his PhD from University of Queensland in 2017. He received bachelor and master degrees in computer science from Huazhong University of Science and Technology in 2011 and 2013, respectively. His research interests include spatio-temporal data management and graph data management.