Privacy-Preserving Bandits

Mohammad Malekzadeh [1] **Dimitrios Athanasakis** [2] **Hamed Haddadi** [2 3] **Benjamin Livshits** [2 3]

Abstract

Contextual bandit algorithms (CBAs) often rely on personal data to provide recommendations. Centralized CBA agents utilize potentially sensitive data from recent interactions to provide personalization to end-users. Keeping the sensitive data locally, by running a local agent on the user's device, protects the user's privacy, however, the agent requires longer to produce useful recommendations, as it does not leverage feedback from other users.

This paper proposes a technique we call *Privacy-Preserving Bandits* (P2B); a system that updates local agents by collecting feedback from other agents in a differentially-private manner. Comparisons of our proposed approach with a non-private, as well as a fully-private (local) system, show competitive performance on both synthetic benchmarks and real-world data. Specifically, we observed only a decrease of 2.6% and 3.6% in multi-label classification accuracy, and a CTR increase of 0.0025 in online advertising for a privacy budget $\epsilon \approx 0.693$. These results suggest P2B is an effective approach to challenges arising in on-device privacy-preserving personalization.

1 Introduction

Personalization is the practice of tailoring a service to individual users by leveraging their interests, profile, and content-related information. For example, a personalized news service would learn to recommend news articles based on past articles the user has interacted with (Li et al., 2010), or a badge achievement system can improve the user engagement by offering gamified activities that are close to the user's interests (Gharibi & Malekzadeh, 2017). Contextual Bandit Algorithms (Li et al., 2010; Ronen et al., 2016) are frequently the workhorse of such personalized services. CBAs improve the quality of recommendations by dynamically adapting to users' interests and requirements. The overall goal of the *agent* in this setting is to learn a policy that maximizes users engagement through time. In doing so, the agent collects potentially sensitive information about a user's interests and past interactions, a fact that raises privacy concerns.

On-device recommendation can address privacy concerns, arising from the processing of users' past interactions, by storing and processing any user feedback locally. As no personal data leaves the user's device, this approach naturally maintains a user's privacy[1]. However, the on-device approach is detrimental to personalization, as it fails to incorporate useful information gleaned from other users, limiting its utility in making new recommendations and leading to the cold-start problem, where the quality of initial recommendations is often insufficient for real-life deployment.

Techniques for privacy-preserving data analysis such as the Encode-Shuffle-Analyze (ESA) scheme implemented in PROCHLO (Bittau et al., 2017; Erlingsson et al., 2020) promise to safeguard user's privacy, while preserving the quality of resulting recommendations, through a combination of cryptographic, trusted hardware, and statistical techniques. In this paper, we explore the question of how the ESA approach can be used in a distributed personalization system to balance the quality of the received recommendations with maintaining the privacy of users.

We propose P2B—Privacy-Preserving Bandits—a system where individual agents running locally on users' devices are able to contribute useful feedback to other agents through centralized model updates while providing differential privacy guarantees (Gehrke et al., 2012; Dwork & Roth, 2013). To achieve this, P2B combines a CBA agent running locally on a user's device with a privacy-preserving data collection scheme similar to ESA.

Specifically, this paper makes the following contributions:

- **Context Encoding**. We propose simple methods of efficiently encoding feedback instances of a CBA on the user's device. The encoding process combines simple clustering algorithms, such as k-means, with the favorable spatial structure of normalized contexts. We study the effects of this structure on both privacy

[1] Queen Mary University of London (this work was done during an internship at Brave Software), [2] Brave Software, [3] Imperial College London. Correspondence to: Mohammad Malekzadeh <m.malekzadeh@qmul.ac.uk>.

Proceedings of the 3^{rd} MLSys Conference, Austin, TX, USA, 2020. Copyright 2020 by the author(s).

[1] In a practical setting, the service maintainer still needs to do extra work to eliminate potential side channels attacks.

and utility, and experimentally demonstrate how this encoding approach is competitive with methods that do not take privacy under consideration.

- **Privacy Analysis**. We examine the overall trade-offs between recommendation quality and privacy loss by performing a differential privacy analysis of our proposal according to the crowd-blending privacy model (Gehrke et al., 2012). Our analysis proves that P2B results in a small ϵ value for differential privacy, which can be directly quantified from the probability of an agent participating in the data collection mechanism. This demonstrates mathematically that P2B provides a concrete and desirable privacy guarantee.

- **Experimental Results**. We construct a testbed for the evaluation of our approach where our proposal is evaluated on synthetic and real-world data. We include results on a synthetic benchmark, multi-label classification, and online advertising data. Our results experimentally demonstrate that P2B remains competitive in terms of predictive utility with approaches that provide no privacy protections. At the same time, it substantially outperforms on-device cold-start models that do not share data[2].

2 Background

Contextual bandit algorithms present a principled approach that addresses the exploration-exploitation dilemma in dynamic environments, while utilizing additional, contextual information. In the contextual bandit setting, at time $t' < t$ the agent selects an action $a_{t'} \in \{1, 2, \dots, |A|\}$ based on the observed d-dimensional context vector $X_{t'}$ at that time. The agent then obtains the reward $r_{t',a} \in \{0, 1\}$ associated with the selected action, without observing the rewards associated with alternative actions.

Upper confidence bound (UCB) method computes upper bounds on the plausible rewards of each action and consequently selects the action with the highest bound. In the implemented P2B, we use *LinUCB* (Chu et al., 2011), which computes UCB based on a linear combination of rewards encountered on previous rounds to propose the next action. In LinUCB the exploration-exploitation actions depend on $\alpha \geq 0$, which is the parameter controlling that trade-off.

In the following we provide some preliminary material on privacy preserving data release mechanisms in relation to personalization with contextual bandit algorithms.

Table 1. Notation used in this paper.

A	set of available actions;
$a_{t'} \in A$	chosen action at time $t' < t$;
$d \in \mathbb{N}$	size of the context vector;
$k \in \mathbb{N}$	number of possible *encoded* contexts.
$n \in \mathbb{N}$	number of possible *original* contexts;
$l \in \mathbb{N}$	crowd-blending threshold;
$p \in [0, 1]$	probability of pre-sampling;
$q \in \mathbb{N}$	precision of digits after the decimal point;
$r_{t',a} \in \{0, 1\}$	reward associated to an action $a_{t'}$;
$t \in \mathbb{N}$	number of interactions with user;
$u \in \mathbb{N}$	number of users;
$X_{t'} \in \mathbb{R}^d$	d-dimensional context vector at time $t' < t$;
$\mathbf{X}$	dataset of context vectors;
$y_{t'} \in \mathbb{N}$	encoded context vector associated to an $X_{t'}$;
$z \in \mathbb{R}$	random Gaussian noise;
$\alpha \in \mathbb{R}^+$	exploration-exploitation ratio in LinUCB;
$\epsilon, \delta \in \mathbb{R}^+$	parameters of differentially privacy.

2.1 Differential Privacy Preliminaries

In the differential privacy framework (Dwork & Roth, 2013) a data sharing mechanism violates its users privacy if data analysis can reveal if a user is included in the data with a degree of confidence higher than the mechanism's bound.

Definition 1: Differentially-Private Data Sharing. Given $\epsilon, \delta \geq 0$, we say a data sharing mechanism $\mathcal{M}$ satisfies $(\epsilon, \delta)-$differential privacy if for all pairs of neighbor datasets of context vectors $\mathbf{X}$ and $\mathbf{X}'$ differing in only one context vector X, such that $\mathbf{X} = \mathbf{X}' \cup \{X\}$, and for all $R \subset Range(\mathcal{M})$,

$$Pr[\mathcal{M}(\mathbf{X}) \in R] \leq e^{\epsilon} Pr[\mathcal{M}(\mathbf{X}') \in R] + \delta$$

Intuitively, a differentially-private $\mathcal{M}$ controls the ability of any attacker to distinguish whether or not a specific user's data is used in the computation, holding the participation of all other users fixed (Kifer et al., 2020).

2.2 Crowd-blending

Crowd-blending privacy model (Gehrke et al., 2012) is a relaxation of differential privacy. In the crowd-blending framework a user blends with a number $l \geq 1$ of users in a way that replacing the user's data with any other individual in this crowd does not alter the results of statistical queries. A necessary condition for this approach is the existence of $l - 1$ other users that blend with the user. If the data does not contain a sufficient number of other individuals the mechanism essentially ignores the user's data. Consequently an *encoding* mechanism $\mathcal{M}(\cdot)$ that satisfies crowd-blending privacy (Gehrke et al., 2012) can be formalized as follows:

Definition 2: Crowd-Blending Encoding. Given $l \geq 1$, we say an encoding mechanism $\mathcal{M}$ satisfies $(l, \bar{\epsilon} = 0)-$crowd-blending privacy if for every context vec-

[2] Code and data are available at: `https://github.com/mmalekzadeh/privacy-preserving-bandits`

tor $\mathbf{x}$ and for every dataset of context vectors $\mathbf{X} = \mathbf{X}' \cup \{X\}$ we have

$$\Big|\{y \in \mathcal{M}(\mathbf{X}) : y = \mathcal{M}(\{X\})\}\Big| \geq l \text{ or } \mathcal{M}(\mathbf{X}) = \mathcal{M}(\mathbf{X}'),$$

where $\big|\{\cdot\}\big|$ denotes the size of a set[3].

This means that for every context vector $X_{t'}$, either its encoded value $y_{t'}$ blends in a crowd of $l-1$ other values in the released dataset, or the mechanism does not include $X_{t'}$ at all. In this setting, $\bar{\epsilon} = 0$ means that if $\mathcal{M}(\cdot)$ releases an encoded value $y_{t'}$ for a given $X_{t'}$, it should be exactly the same as other $l-1$ encoded values coming from other $l-1$ context vectors. Every (ϵ, δ)-differentially private mechanism also satisfies $(\mathrm{l}, \epsilon, \delta)$-crowd-blending privacy for every integer $l \geq 1$ (Gehrke et al., 2012).

An important aspect of crowd-blending privacy is that if a random *pre-sampling* is used during data collection, before applying the crowd-blending on the collected data, the combined pipeline provides zero-knowledge (Gehrke et al., 2011) privacy and thus differential privacy (Dwork & Roth, 2013) as well. This intuitively says that if my data blends in a crowd of l other people, then other people similar to me can easily take the place of me. Therefore, my data being sampled or not has a negligible effect on the final output of the combined mechanism. P2B utilizes this property.

2.3 Differentially-private Data Collection

Privacy-preserving data collection has numerous applications. To collect browser settings data, Google has built RAPPOR into the Chrome browser (Erlingsson et al., 2014). RAPPOR hashes textual data into Bloom filters using a specific hash function. It then randomizes the Bloom filter, that is a binary vector, and uses it as the permanent data to generate instantaneous randomized response for sharing with the server. RAPPOR can be used to collect some aggregated statistics from all users. For example, having a huge number of users, it can estimate the most frequent homepage URLs. However, since the shared data is only useful for estimating aggregated statistics like mean or frequency, the utility of every single shared data for training a model is often too low — the accuracy of the private data becomes unacceptable even with a large number of users.

As an extension to RAPPOR, the ESA architecture (Bittau et al., 2017; Erlingsson et al., 2020) adds two more layers to the LDP layer, called shuffler and analyzer, that are aimed to obscure the identity of the data owner through oblivious shuffling with trusted hardware. The Shuffler in the ESA architecture eliminates all metadata attached to the users' reports to eliminate the possibility of linking a data report to a single user during data collection. However, if users do not want to trust any other party, the provided privacy will be the same as RAPPOR.

3 Methodology

In P2B, every user runs their own CBA agent that works independently of any other agents[4]. At time t' the agent observes the current context $X_{t'}$, which represents recent user's activities (*e.g.* recent web browsing history). Based on $X_{t'}$, the agent proposes an action $a_{t'}$, and consequently observes the reward $r_{t',a}$ associated with the action (*e.g.* proposing a news article and observing the user's reaction to that article). As the interaction proceeds locally, we refer to agents running on a user's device as *local agents*. The agent may periodically elect to send some information about an observed interaction to a data collection server. Using the collected data, the server updates a central model that is then propagated back to the users.

Figure 1 summarizes the overall architecture of the proposed framework. The rest of this section presents P2B's operation and details its various components. Section 4 describes how P2B satisfies differential privacy, and Section 5 experimentally demonstrates how this approach improves the performance of local agents.

3.1 Randomized Data Reporting

Local agents participate in P2B through a randomized participation mechanism. After some interactions with the user, $t \geq 1$, the local agent may randomly construct a payload, containing an encoded instance of interaction data, with probability p. Randomized participation is a crucial step in two ways. First, it raises the difficulty of re-identification attacks by randomizing the timing of recorded interactions. More importantly, the participation probability p as a source of randomness, has a direct effect on the differential privacy parameters ϵ and δ we will establish in Section 4. Briefly, by choosing the appropriate participation probability, one can achieve any level of desired privacy guarantee in P2B.

3.2 Encoding

The agent encodes an instance of the context prior to data transmission. The encoding step acts as a function that maps a d-dimensional context vector X into a code $y \in \{0, 1, \dots, k-1\}$, where k is the total number of encoded contexts. Agents encode a normalized context vector of fixed precision that uses q digits for each entry in the vector. An example of this representation approach is a normalized histogram, where entries in the histogram sum to 1 and are represented to a precision of q decimal digits. This combination of normalization and finite precision has two important characteristics.

[3] Note that $\mathbf{X}'$ can be any context dataset not including X.

[4] Note that *user* is an individual and *agent* is an on-device algorithm that serves a user.

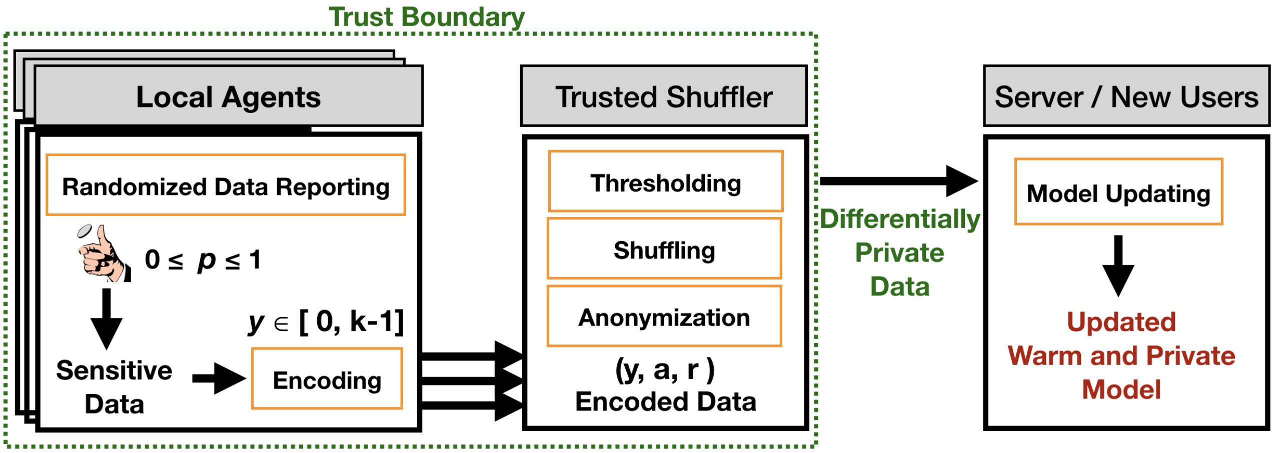

Figure 1. System architecture for P2B.

First, it is possible to precisely enumerate all possible contexts according to the *stars and bars* pattern in combinatorics (Benjamin & Quinn, 2003). Specifically the cardinality n of the set of normalized context vectors $X \in \mathbf{X}$, using a finite precision of q decimal digits is

$$n = \binom{10^q + d - 1}{d - 1}. \qquad (1)$$

Secondly, sample points in the normalized vector space are distributed uniformly in a grid shape. Given that the agent tends to propose similar actions for similar contexts, neighboring context vectors can be encoded into the same context code y. While this approach may appear to be limiting, it generalizes to other instances where the context vectors are bounded and exhibit some degree of clustered structure.

Figure 2, provides a concrete example of the encoding process on a 3-dimensional vector space being encoded in $k = 6$ different codes. The cardinality of the encoding, n, is important in establishing a desired utility-privacy trade off. The encoding algorithm can be chosen depending on application requirements, but we limited our experimental evaluation to the simple k-means clustering (Sculley, 2010).

After electing to participate and encoding an instance of a user interaction, the agent transmits the data in the form of the tuple $(y_{t'}, a_{t'}, r_{t',a})$ to the *shuffler*.

3.3 Shuffler

The trusted shuffler is a critical part of every ESA architecture (Bittau et al., 2017; Erlingsson et al., 2020), and in P2B it is necessary for ensuring anonymization and crowd-blending. Following the same PROCHLO implementation (Bittau et al., 2017), the shuffler operates in a secure enclave on trusted hardware and performs three tasks:

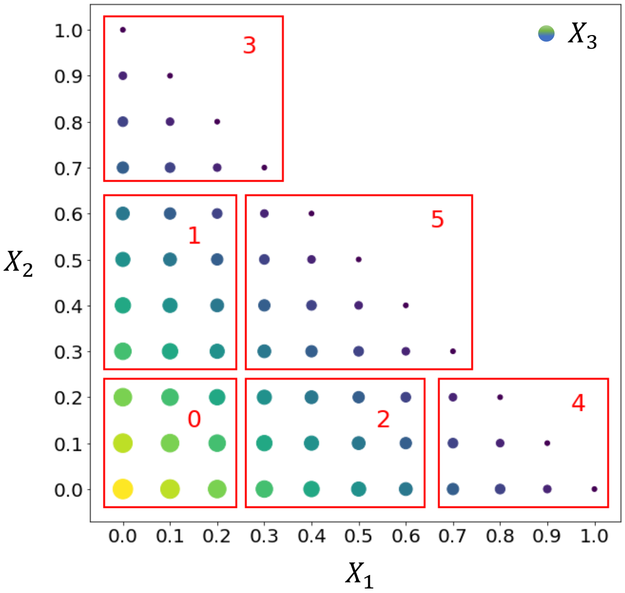

Figure 2. Normalized vector space of $X = (X_1, X_2, X_3)$ with $q = 1$ and cardinality $n = 66$. In this setting, we have normalized vectors such that $X_1 + X_2 + X_3 = 1$. Thus, size of the circles shows the value of X_3. Rectangles show a potential encoding of the vector space with $k = 6$ and minimum cluster size $l = 9$.

1. **Anonymization**: eliminating all the received metadata (*e.g.* IP address) originating from local agents.
2. **Shuffling**: gathering tuples received from several local agents into batches and shuffling their order. This helps to prevent side-channel attacks (*e.g.* timing attack).
3. **Thresholding**: removing tuples that their encoded context vector frequency in the batch is less than a defined *threshold*. This threshold plays an important role in the achieved differential privacy bound.

After performing these three operations, the shuffler sends the refined batch to the *server* for updating the model. Upon receiving the new batch of training data, the server updates the global model and distributes it to local agents that request it. More importantly, this global model is used as a warm-start by new local agents that are joining to the system to help in preventing the cold-start problem.

4 Privacy Analysis

This section analyzes how P2B ensures differential privacy through a combination of pre-sampling and crowd-blending. As context vectors are multi-dimensional real-valued vectors, we assume a context vector in its original form can be uniquely assigned to a specific user. P2B assumes no prior on the possible values for a context vector, meaning context vectors are coming from a uniform distribution over the underlying vector space. With these assumptions and the provided zero-knowledge privacy, P2B can resist strong adversaries with any kind of prior knowledge or side information.

For the n different context vectors in Equation (1), the optimal encoder (Section 3.2) encodes every n/k contexts into one of the possible k codes. Consequently, when a total number of u users participate in P2B to sends a tuple to the server, the optimal encoder satisfies crowd-blending privacy with $l = u/k$. In the case of a suboptimal encoder, we consider l as the size of the smallest cluster in the vector space. Furthermore, situations where the number of users is small, leading to a small l, can be addressed by adjusting the shufflers *threshold* to reach the desired l. Essentially, l can always be matched to the shuffler's threshold.

Each user randomly participates in data sharing with probability p (Section 3.1), and then encoding the pre-sampled data tuple. Following (Gehrke et al., 2012), the combination of (i) pre-sampling with probability p and (ii) $(l, \bar{\epsilon})$−crowd-blending, leads to a differentially private mechanism with

$$\epsilon = \ln\Big(p \cdot (\frac{2-p}{1-p} \cdot e^{\bar{\epsilon}}) + (1-p)\Big) \qquad (2)$$

and

$$\delta = e^{-\Omega(l\cdot(1-p)^2)}. \qquad (3)$$

Here Ω is a constant that can be calculated based on the analysis provided by (Gehrke et al., 2012).

Our encoding scheme provides an $\bar{\epsilon} = 0$ for crowd-blending, as the encoded values for all the members of a crowd is exactly the same. As a consequence, the ϵ parameter of the differential privacy of the entire data sharing mechanism depends entirely on the probability p of participation as (Figure 3):

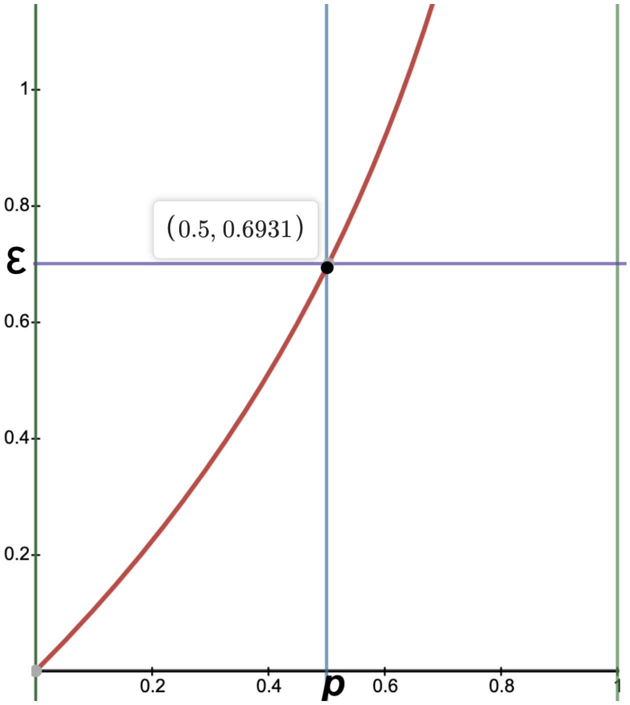

Figure 3. ϵ as a result of the probability of participating in the scheme p.

$$\epsilon = \ln\Big(p \cdot (\frac{2-p}{1-p}) + (1-p)\Big).$$

For example by trading half of the potential data ($p = 0.5$) P2B achieves an $\epsilon \approx\ 0.693$ that is a strong privacy guarantee. On the other hand, δ depends on both p and l. To understand the effect of δ, Dwork *et al.* (Dwork & Roth, 2013) prove that an (ϵ, δ)-differentially private mechanism ensures that for all neighboring datasets, the absolute value of the privacy loss will be bounded by ϵ with probability at least $1 - \delta$. Therefore, by linearly increasing the crowd-blending parameter l, we can exponentially reduce the δ parameter.

Our illustrations here provide concrete settings and parameters that show how P2B satisfies the requirements of the differential privacy model proposed in (Gehrke et al., 2012). We refer the interested reader to the rigorous analysis and proofs provided in the original paper (Gehrke et al., 2012).

5 Exprimental Evaluation

This section explores P2B's privacy-utility trade-offs compared to (i) competing non-private and (ii) completely private approaches. The experimental evaluation assumes the standard bandit settings, where the local agent learns a policy based on the contextual Linear Upper Confidence Bound algorithm (LinUCB) (Chu et al., 2011; Li et al., 2010). For simplicity, throughout the experiments, the probability p of randomized transmission by a local agent was set to $p = 0.5$, the rounding parameter q of the encoder was set to $q = 1$, and the parameter α for LinUCB was set to $\alpha = 1$, meaning that the local agent is equally likely to propose an exploration or exploitation action. The experiments compare the performance of the following three settings:

Cold. The local agent learns a policy without any communication to the server at any point. As there is no communication, this provides full privacy, but each agent has to learn a policy from a cold-start.

Warm and Non-Private. In this setting, local agents communicate the observed context to the server in its original form. Thus, other agents are able to initialize their policy with a model received from the server and start adapting it to their local environment. This is called warm and non-private start, and represents the other end of the privacy/utility spectrum with no privacy guarantee afforded to the users.

Warm and Private In this setting, local agents communicate with the server using P2B. Once more, other agents initialize their internal policy with an updated model received from the server and start to adapt it to their local environment. We term this a warm and private start, and the provided privacy guarantees function according to the analysis in section 4.

These approaches are evaluated on synthetic benchmarks, two multi-label classification datasets, and an online advertising dataset.

5.1 Synthetic Preference Benchmark

These benchmarks consider the setting where there's a stochastic function $\mathcal{F}$ that relates context vectors with the probability of a proposed action receiving a reward. Specifically, $\mathcal{F}$ is the scaled softmax output of a matrix-vector product of the user contexts with a randomly generated weight matrix W:

$$\mathcal{F}(X) = \mathsf{softmax}(W \cdot X + b) \in [0, 1]^{|A|}. \tag{4}$$

For the specific implementation in this section, we use the default weight initializer of TensorFlow (Abadi et al., 2016a) to set W and b, and then keep them fixed through the experiment. In the LinUCB algorithm, we set the mean reward $\bar{r}_a$ for every action a_t as

$$\bar{r}_a = \beta \mathcal{F}_a(X) + z,$$

where $\mathcal{F}_a$ is the a-th component of the softmax output in Equation 4, β is a scaling factor with $0 \leq \beta \leq 1$, and z is a random Gaussian noise $z \sim \mathcal{N}(0, \sigma^2)$. Hence, to create an instance of a synthetic interaction between user and agent, we first randomly and uniformly generate a context vector $\mathbf{x}$. Then having $\mathbf{x}$ and using $\mathcal{F}$, for each possible action a we calculate the probable reward r. For time $t' < t$, this process results to a synthetic data tuple $(X_{t'}, a_{t'}, r_{t',a})$.

For all synthetic benchmarks the parameters are set as follows. The scaling factor for the preference function was fixed to $\beta = 0.1$ and the variance of the Gaussian noise $\sigma^2 = 0.01$. In terms of local agent settings, the agents use $k = 2^{10}$ codes for encoding purposes and observe $t = 10$ local interactions before randomly transmitting an instance with probability $p = 0.5$. We varied the number of dimensions d of the context vector in the range of 6 to 20, and the number of actions in the range of 10 to 50. We observe the average reward in each setting as the user population u grows from 10^3 to 10^6.

Our results in Figure 4 indicate that for a small number of interactions the cold-start local model fails to learn a useful policy. By contrast, the warm models substantially improve as more user data becomes available. Utilizing prior interaction data in this setting, more than doubles the effectiveness of the learned policies, even for relatively small user populations. Overall, the non-private agents have a performance advantage, with the private version trailing.

Figure 5 illustrates how the dimensionality of the context vector affects the agent's expected reward. By increasing d from 6 to 20 the average reward decreases as agents spend more time exploring the larger context space. P2B remains competitive with its non-private counterparts, and on occasion outperforms them, especially for low-dimensional context settings. The number of actions has a similar effect to the dimensionality of the context, as agents have to spend more time exploring suboptimal actions. Once more, the results of Figure 4 indicate that this is the case experimentally.

The number of users is pivotal in the trade-off between privacy and utility. For simplicity, and in order to avoid unnecessary computation, experiments fix $p = 0.5$. However, the expected reward for other p values can be simply deduced from the current experiments. For example, if p changes from 0.5 to 0.25 (meaning ϵ is changed from 0.69 to 0.28) P2B would require $100K$ users rather than $50K$ to achieve the same performance with the new privacy guarantee. In simple terms, to achieve smaller ϵ, a smaller p is required, which consequently means a proportionally increased number of users is necessary to maintain the same performance.

5.2 Multi-Label Classification

These experiments examine the performance of the three settings on multi-label classification with bandit feedback. We consider two datasets, namely (1) MediaMill (Snoek et al., 2006), a video classification dataset including 43,907

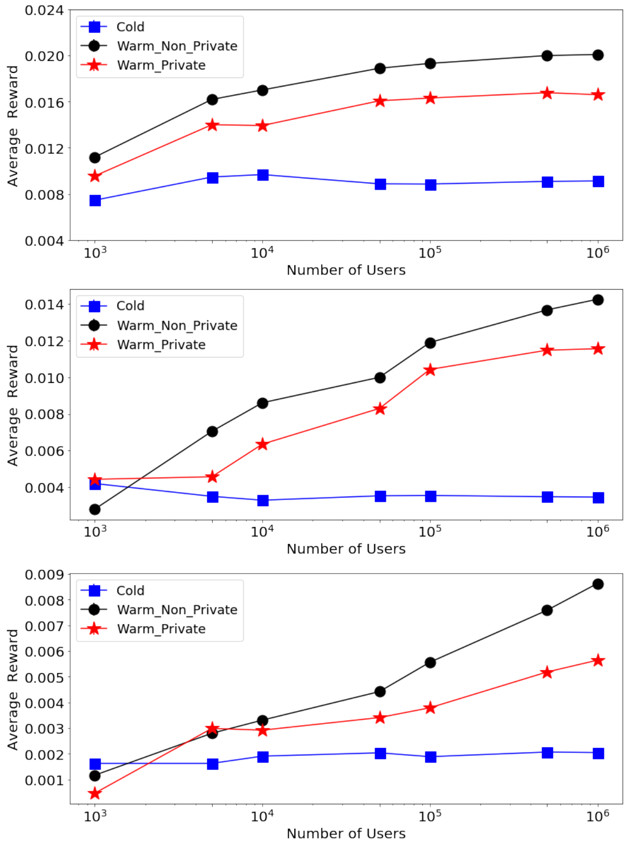

Figure 4. Synthetic benchmarks: (Top) $|A| = 10$, (Middle) $|A| = 20$ (Bottom) $|A| = 50$. For all: $d = 10$ and $t = 10$. The expected reward in this setting has a strong dependence on the number of actions as agents will spend considerable time exploring alternative actions.

instances, 120 extracted features from each video, and 101 possible categories, and (2) a TextMining dataset (Srivastava & Zane-Ulman, 2005) including 28,596 instances, 500 extracted features from each text, and 22 possible categories.

P2B's performance in this setting was evaluated according to the following setting. We consider a fixed number of local agents. Each agent has access to, and is able to interact with a small fraction of the dataset. In particular every agent has access to up to 100 samples, which were randomly selected without replacement from the entire dataset. 70% of agents to participate in P2B and we test the accuracy of the resulting models with the remaining 30%. For both datasets, the local agents use $k = 2^5$ codes for encoding purposes.

This setting is particularly interesting as it allows us to study how the predictive performance of the system changes when the local agents interact more with the user. In terms of relative performance between the different approaches, the results in Figure 6 repeat the findings of the synthetic benchmarks. The non-private warm version is better than the private warm version, which is still better than the cold version that utilizes only local feedback.

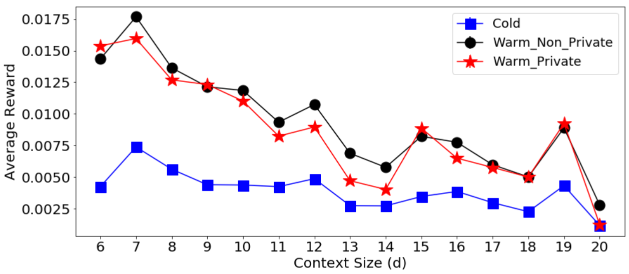

Figure 5. Synthetic benchmarks: $u = 20000$, $|A| = 20$, $t = 20$, and $d = \{6, 7, \dots, 20\}$. As the dimensionality of the context increases the average reward for this settings is reduced as agents spend more time trying to explore their environment.

However, we can also observe that the cold version given enough interactions produces increasingly improving results. The centralized update mechanism tends to have a multiplicative effect, especially when there is little local interaction data before reaching a plateau.

5.3 Online Advertising

Here, we consider an advertisement recommendation scenario where the action is to recommend an ad from one of the existing categories. We use the Criteo dataset from a Kaggle contest[5]. It consists of a portion of Criteo's traffic over a period of 7 days, including 13 numerical features and 26 categorical features. For each record in the dataset there is a label that shows whether the user has clicked on the recommended ad or not. For anonymization purposes, the feature's semantics are unknown and the data publisher has hashed the values of categorical features into 32 bits.

As the exact semantics of the features were not disclosed, we assume that numerical features represent user's context and categorical features correspond to the type of the proposed product. For each sample in the dataset, we hash the values of the 26 categorical features into an integer value. The resulting integer value is then used as possible product category to recommend. The hashing procedure operates as follows: First, the 26 categorical values are reduced into a single hashed value using feature hashing (Weinberger et al., 2009). After hashing, the 40 most frequent hash codes are selected. These are converted into an integer value in the range between 1 to 40, based on their frequency (label 1 shows the most frequent code and so on). Finally, for evaluation we only use data samples having one of this 40 values as the product label and ignore the remaining data.

[5] `https://labs.criteo.com/category/dataset`

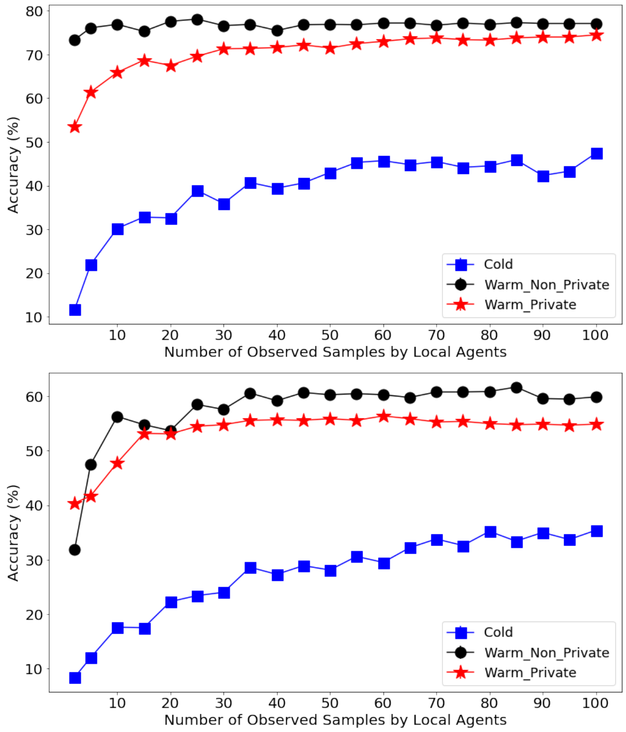

Figure 6. Accuracy in multi-label datasets: (Top) Media-Mill with $d = 20$ and $|A| = 40$ (Bottom) Text-Mining with $d = 20$ and $|A| = 20$. As local agents observe more interactions they obtain better accuracy. This has a multiplicative effect in the distributed settings where agents reach to the plateau much faster.

During evaluation, the local agent observes the values of the numerical features as the context vector, and in response takes one of the $|A| = 40$ possible actions. The agent obtains a reward of 1, if the proposed action matches the logged action in the dataset. The remaining experimental setup for the Criteo dataset is similar to the one used in the multi-label datasets. Specifically, we present experimental results (Figure 7) for d=10 and total number of actions $|A| = 40$. We compare the results for two values of encoding parameter $k = 2^5$ and $k = 2^7$. The sampling probability p remains 0.5, and the shuffling threshold remains 10. This experimental setting has $u = 3000$ agents, and each of the agents accumulates 300 interactions.

The results in this setting are quite surprising as private agents attain a better click-through rate than their non-private counterparts. There is a recent work (Nasr et al., 2018) showing that privacy-preserving training of machine learning models can aid generalization as well as protect privacy. There are some factors we believe help explain these experimental results.

Private agents use the encoded value as the context. As a result, the number of possible contexts is much smaller compared to using the original d-dimensional context vector. As contextual bandits need to balance exploration with exploitation, especially in the early stages, a bandit with a smaller context size can quickly reach better results. Furthermore, P2B clusters similar context vectors into the same categories and this also helps a private bandit to better act in similar situations. This is an effect that also can be seen for the higher dimensional contexts of the synthetic benchmarks in Figure 5.

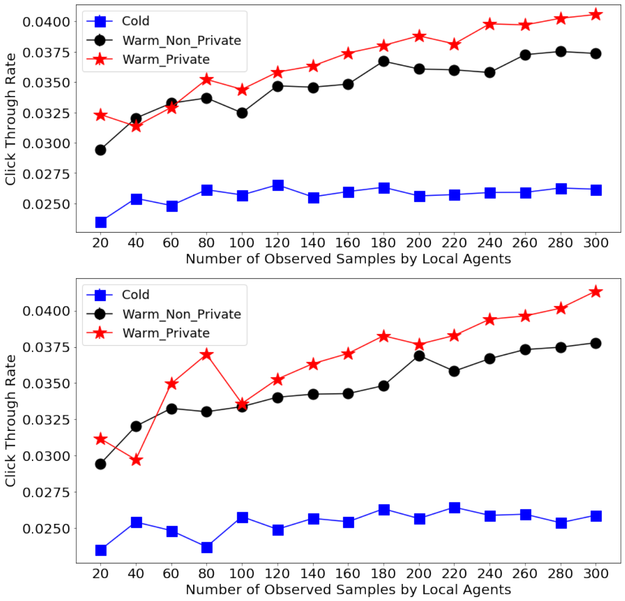

Figure 7. Criteo results. $d = 10$, $|A| = 40$, (Top) $k = 2^5$, and (Bottom) $k = 2^7$. The private and non-private agents obtain similar performances for low numbers of local interactions. As the number of local interactions increase the private agents perform better than their non-private counterparts.

Once more, the number of users plays a critical role in establishing an acceptable trade-off between utility and privacy. As Section 4 explains, δ in Equation 3 can be interpreted as the probability that the worst-case happens in differentially private data sharing. Using a larger k in a system with a small number of users u, limits the crowd blending parameter l and consequently leads to undesired potential values for δ. Unfortunately, the number of users for real-world datasets used in this paper is limited posing constraints on the range of parameters in the experimental evaluation.

6 Related Work

User data fuels a variety of useful machine learning applications, but at the same time comes with a number of ethical and legal considerations. The quest to balance the seemingly conflicting concerns of privacy and utility have shaped the resurgence of distributed machine learning approaches that safeguard the privacy of user data. This section compares

P2B to some recent proposals in this area.

On-device inference ensures that data does not need to leave the device for inference, which is an important step towards protecting privacy. Federated learning techniques extend this approach to doing on device training through the idea of "bringing the code to the data, instead of the data to the code". (Bonawitz et al.) proposes and evaluates a high-level system to perform federated learning across devices with varying availability. Their proposal relies on secure aggregation (Bonawitz et al., 2017; Konečný et al., 2016) to ensure that user's data remain encrypted even in the server's memory.

Compared to P2B, federated learning is substantially more complicated in terms of orchestration. Relying on a local model training procedure, it puts some overhead on the user's device and there is substantial communication complexity involved in communicating model updates and training batching. By contrast, P2B puts almost no overhead on user's devices as they only need to run a distance preserving encoding algorithm (Sculley, 2010; Aghasaryan et al., 2013) that has the $O(kd)$ complexity at the inference time, and any data exchange happens opportunistically, without requiring any synchronization between clients.

Moreover, the scope of P2B is personalization. This means P2B is concerned with developing individual models adapted to individual users' preferences, whereas federated learning is primarily concerned with improving a global model. P2B addresses a personalization problem which is closer to the reinforcement learning setting, than supervised learning. In our setting, feedback is partial, and there can be more than one valid responses for the exact same context. This makes a direct utility comparison with supervised learning methods such as federated learning (Konečný et al., 2016), or differentially private deep learning (Abadi et al., 2016b) infeasible.

P2B, however, provides an interesting performance in terms of complexity. It leverages existing PROCHLO infrastructure (Bittau et al., 2017) without needing modifications to the aggregate privatization layer. As illustrated in Figure 1, the total scope of changes required for P2B to work is only a clustering lookup table and procedure on the client, allowing P2B to use an existing ESA deployment without modification, and the central server only needs to do simple contextual bandit updates.

The "Draw and Discard" machine learning system described in (Pihur et al., 2018) is an interesting combination of the approach of bringing the code to the data with differential privacy guarantees. Their proposed system encompasses decentralized model training for generalized linear models. It maintains k instances of the model at the server side. The draw and discard approach consists of two components: On the server side, it sends a randomly chosen model instance to the client. Upon receiving an updated model instance, the server randomly replaces it with one of the k existing instances. On the user side, users update the linear model on their data, and add Laplacian noise to the model parameters to satisfy differential privacy.

Privacy analysis of "Draw and Discard" is in the feature-level that assumes features of the model are all uncorrelated. As a consequence, for achieving model-level privacy, where most features are correlated to each other, substantially more noise needs to be added to the updated model before sending it to the server. P2B does not make any assumption on the correlation among context vector features and it provides a constant and similar privacy guarantee across all the users. Our experiments only consider situations where users participate by sending only one data tuple. However, in the case of collecting m data tuple from each user, because of the composition property of the provided differential privacy (Dwork & Roth, 2013), P2B still guarantees ($m\epsilon$)-differential privacy.

Shariff *et al.* (Shariff & Sheffet, 2018) discuss that adhering to the standard notion of differential privacy needs to ignore the context and thus incur linear regret. Thus, they use a relaxed notion of DP, called "joint differential privacy", that allows to use the non-privatized data at time t' , while guaranteeing that all interactions with all other users at timepoints $t'' > t'$ have very limited dependence on that user data at time t'. Our approach guarantees differential privacy to all users, independent of time of participation in P2B.

Basu *et al.* (Basu et al., 2019) argue that if a centralized bandit algorithm only observes a privatized version of the sequence of user's actions and responses, based on ϵ-LDP, the algorithm's *regret* scales as a multiplicative factor ϵ. Other LDP mechanisms proposed for bandits algorithms (Tossou & Dimitrakakis, 2017; Gajane et al., 2018) consider a situations where only responses are considered to be private without incorporating contextual information in the learning process.

Although individual components of P2B, namely contextual bandit algorithms and differentially private mechanisms for data collection, have been widely studied, what P2B achieves is combining these components in a novel way that allows controlling the trade-off between overall utility and privacy, relying solely on two straightforward parameters: (i) the probability of participation in the randomized data collection, and (ii) the chosen threshold for the crowd-blending.

7 Conclusions

This paper presents P2B, a privacy-preserving approach for machine learning with contextual bandits. We show experimentally that standalone agents trained on an individual's data require a substantial amount of time to learn a useful recommendation policy. Our experiments show that sharing data between agents can significantly reduce the number of required local interactions in order to reach a useful local policy.

We introduced a simple distributed system for updating agents based on user interactions. In a series of experiments, P2B shows considerable improvement for increasing numbers of users and local interactions. With regards to P2B's privacy, experiments demonstrate the clustering-based encoding scheme is effective in encoding interactions. In addition, all the experiments relied on a sampling probability $p = 0.5$. This results in a very competitive privacy budget of $\epsilon \approx 0.693$.

Given that P2B essentially discards half of the data for the sake of privacy, the system performs surprisingly well. In the synthetic benchmarks, P2B traces a similar trend to its non-private counterpart. While for large populations the non-private agents have an advantage, there are more than a few cases where their private counterparts are competitive and even outperform them. Additionally, as the multi-label classification experiments (Figure 6) indicate, the performance gap between the non-private and differentially-private settings seems to be shrinking to within 3.6% and 2.6% for the text mining and MediaMill tasks respectively. P2B's performance on the Criteo dataset is a rare instance where a privacy preserving regime actually improves generalization. The resulting CTR difference of 0.0025 in favor or the privacy preserving approach is somewhat surprising. We do provide reasons as to why this is not completely unexpected in our evaluation of results.

Overall, P2B represents a simple approach to perform privacy-preserving personalization. Our results indicate that P2B becomes a particularly viable option for settings where large user populations participate in a large amount of local interactions, where the performance penalty for privacy is vanishingly small. As future work, we aim to study the behavior of more encoding approaches as well as their interplay with alternative contextual bandit algorithms.

Acknowledgements

Mohammad Malekzadeh wishes to thank the Life Sciences Institute at Queen Mary University of London and professor Andrea Caballero for their support through the project. Hamed Haddadi was partially supported by the EPSRC Databox grant (EP/N028260/1). We would like to thank Dr. Antonio Nappa for his constructive feedback and insights. We would also like to thank anonymous reviewers for their helpful comments and suggestions.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016a.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016b.

Aghasaryan, A., Bouzid, M., Kostadinov, D., Kothari, M., and Nandi, A. On the use of LSH for privacy preserving personalization. In *Proceedings of 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom*, 2013.

Basu, D., Dimitrakakis, C., and Tossou, A. Differential privacy for multi-armed bandits: What is it and what is its cost? *arXiv preprint arXiv:1905.12298*, 2019.

Benjamin, A. T. and Quinn, J. J. *Proofs that really count: the art of combinatorial proof.* Number 27. MAA, 2003.

Bittau, A., Erlingsson, U., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 441–459, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. In *SysML 2019*.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical Secure Aggregation for Privacy-Preserving Machine Learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17*, pp. 1175–1191, 2017.

Chu, W., Li, L., Reyzin, L., and Schapire, R. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Dwork, C. and Roth, A. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013.

Erlingsson, Ú., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, 2014.

Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Song, S., Talwar, K., and Thakurta, A. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618*, 2020.

Gajane, P., Urvoy, T., and Kaufmann, E. Corrupt Bandits for Preserving Local Privacy. In *Algorithmic Learning Theory*, pp. 387–412, 2018.

Gehrke, J., Lui, E., and Pass, R. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *Theory of cryptography conference*, pp. 432–449. Springer, 2011.

Gehrke, J., Hay, M., Lui, E., and Pass, R. Crowd-blending privacy. In *Annual Cryptology Conference*, pp. 479–496. Springer, 2012.

Gharibi, R. and Malekzadeh, M. Gamified incentives: A badge recommendation model to improve user engagement in social networking websites. *International Journal of Advanced Computer Science and Applications*, 2017.

Kifer, D., Messing, S., Roth, A., Thakurta, A., and Zhang, D. Guidelines for implementing and auditing differentially private systems. *arXiv preprint arXiv:2002.04049*, 2020.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

Nasr, M., Shokri, R., and Houmansadr, A. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2018.

Pihur, V., Korolova, A., Liu, F., Sankuratripati, S., Yung, M., Huang, D., and Zeng, R. Differentially-private" draw and discard" machine learning. *arXiv preprint arXiv:1807.04369*, 2018.

Ronen, R., Yom-Tov, E., and Lavee, G. Recommendations meet web browsing: Enhancing collaborative filtering using internet browsing logs. *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, pp. 1230–1238, 2016.

Sculley, D. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178, 2010.

Shariff, R. and Sheffet, O. Differentially private contextual linear bandits. In *Advances in Neural Information Processing Systems*, pp. 4296–4306, 2018.

Snoek, C. G., Worring, M., Van Gemert, J. C., Geusebroek, J. M., and Smeulders, A. W. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th Annual ACM International Conference on Multimedia, MM 2006*, 2006.

Srivastava, A. N. and Zane-Ulman, B. Discovering recurring anomalies in text reports regarding complex space systems. In *IEEE Aerospace Conference*, 2005.

Tossou, A. C. Y. and Dimitrakakis, C. Achieving privacy in the adversarial multi-armed bandit. 2017.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference On Machine Learning, ICML*, 2009.

A Artifact Appendix

A.1 Abstract

This section outlines the requisite steps to reproduce the experimental results reported in Section 5. It also provides pointers to the source code and datasets used for all of the experiments in a GitHub repository.

A.2 Artifact check-list (meta-information)

- **Algorithm:** Privacy-Preserving Bandits.
- **Program:** Python 3.6 and above.
- **Data set:** Synthetic, MediaMill, TextMining, Criteo (see Section A.3.4).
- **Hardware:** We used a 2.3 GHz 8-Core Intel Core i9 with 16 GB RAM. However, any machine that can run Python 3.6 and above is suitable.
- **Metrics:** Average Reward, Accuracy, and Click-Through Rate for synthetic, multi-label and online advertising datasets, respectively.
- **Output:** Numerical results and their corresponding graphical plots.
- **Experiments:** See Section A.5. More details are reported in Section 5.
- **How much disk space required (approximately)?:** 5GB.
- **How much time is needed to prepare workflow (approximately)?:** About 30 minutes to set up the environment for the first time.
- **How much time is needed to complete experiments (approximately)?:** About 15 minutes for the small example provided on Synthetic data. To get all results reported for all experiments, one may need about 24 hours.
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** MIT license.
- **Data licenses (if publicly available)?:** MIT license.

A.3 Description

A.3.1 Availability

The source code for running all experiments is available in a GitHub repository:

`https://github.com/mmalekzadeh/privacy-preserving-bandits`

The public DOI:

`https://doi.org/10.5281/zenodo.3685952`

The artifact evaluated by MLSys 2020 AE chairs: `https://codereef.ai/portal/c/cr-lib/e5a9d2feea4bb3fa`

A.3.2 Hardware dependencies

Any hardware that can run Python 3.6 programs and have enough memory to host the required datasets. We used a 2.3 GHz 8-Core Intel Core i9 with 16 GB RAM.

A.3.3 Software dependencies

The following python libraries need to be installed: `iteround`, `pairing`, `scikit-multilearn`, `arff`, `category_encoders`, `matplotlib`, `tensorflow`, `keras`.

A.3.4 Data sets

The following datasets has been used:

- MediaMill (Snoek et al., 2006)
- TextMining dataset (Srivastava & Zane-Ulman, 2005)
- Criteo dataset from a Kaggle contest[6].

By running the experiment, MediaMill and TextMining datasets will be automatically downloaded for the first time. For criteo dataset, in the first time, the script `create_datasets.ipynb` should be used. This script is provided in the same repository.

A.4 Installation

A `requirements.txt` is provided in the repository. After installing `Python 3.6` or above, you just need to run the following command:

```
pip install -r requirements.txt
```

Then you can continue with the instructions provided in Section A.5.

Otherwise, you may need to install and run a `Jupyter Notebook`[7] then install these libraries via the code provided in the notebooks:

- `$ pip install iteround`
- `$ pip install pairing`
- `$ pip install scikit-multilearn`
- `$ pip install category_encoders`
- `$ pip install matplotlib`
- `$ pip install tensorflow`
- `$ pip install keras`

[6] `https://labs.criteo.com/category/dataset`

[7] `https://jupyter.org`

A.5 Evaluation and expected result

The general workflow for reproducing the experimental results reported in Section 5 consists of two steps: (i) building an encoder, and subsequently (ii) running the experiments for each dataset.

All the files necessary to reproduce the results are available as Jupyter Notebooks. Specifically, the GitHub repo contains one Jupyter notebook for building an encoder and a separate Jupyter Notebook explaining step-by-step how to run code and reproduce the results for each of the experiments in Section 5.

1. `build_an_encoder.ipynb` Trains a k-means clustering (Sculley, 2010) based encoder. Depending on the application domain, parameters can be chosen to get the desired *Encoder* that is consequently used in other experiments. In this file, function `build_hasher(context_size, bin_size, dec_digits)` gets the required parameters, d, k, and q explained in Section 3.2, to build an *Encoder*.
2. `a_synthetic_exp.ipynb` reproduces the average reward results for synthetic datasets reported in Section 5.1.
3. `mlc_exp.ipynb` reproduces the classification accuracy results for multi-label classification reported in Section 5.2.
4. `criteo_exp.ipynb` reproduces the click-through rate results for online advertising reported in Section 5.3.
5. `a_synthetic_figure_5.ipynb` reproduces the results depicted in Figure 5.
6. `to_plot_the_results.ipynb` contains a sample code for plotting results. Note that, one needs to first run a desired experiment, then use this code to plot the results that have been saved during the experiment.

The experiments can be reproduced and even extended by using the following two routines:

1. `build_hasher(context_size, bin_size, dec_digits)` gets the required parameters, d, k, and q explained in Section 3.2, to build an *Encoder*.
2. New experimental setting can be explored by using `Simulation.run_simulation(n_users, n_samples, n_actions, cb_sampling_rate, cb_context_threshold, neg_rew_sam_rate)` that allow setting the desired set of parameters for each experiment.
    - `n_users`: number of users u.
    - `n_samples`: number of samples per user t.
    - `n_actions`: number of actions $|A|$.
    - `cb_sampling_rate`: sampling parameter, p, for crowd blending (Section 4).
    - `cb_context_threshold`: thresholding parameter l (Section 4).
    - `neg_rew_sam_rate`: probability of sharing data if the reward is negative. This is fixed (5%) across all the experiments reported in this paper.

The outcome of running this function is Average Reward, Accuracy, or Click-Through Rate for Synthetic, Multi-Label Classification or Online Advertising datasets, respectively.