A Bandit Approach to Maximum Inner Product Search

Rui Liu
Computer Science and Engineering
University of Michigan, Ann Arbor
ruixliu@umich.edu

Tianyi Wu
Computer Science and Engineering
University of Michigan, Ann Arbor
tianyiwu@umich.edu

Barzan Mozafari
Computer Science and Engineering
University of Michigan, Ann Arbor
mozafari@umich.edu

Abstract

There has been substantial research on sub-linear time approximate algorithms for *Maximum Inner Product Search* (MIPS). To achieve fast query time, state-of-the-art techniques require significant preprocessing, which can be a burden when the number of subsequent queries is not sufficiently large to amortize the cost. Furthermore, existing methods do not have the ability to *directly* control the suboptimality of their approximate results with theoretical guarantees. In this paper, we propose the first approximate algorithm for MIPS that does not require any preprocessing, and allows users to control and bound the suboptimality of the results. We cast MIPS as a Best Arm Identification problem, and introduce a new bandit setting that can fully exploit the special structure of MIPS. Our approach outperforms state-of-the-art methods on both synthetic and real-world datasets.

Introduction

The problem of *Maximum Inner Product Search* (MIPS) has received significant attention in recent years (Yu et al. 2017; Shrivastava and Li 2014; Neyshabur and Srebro 2015) as a key step in many machine learning algorithms and applications. For instance, it appears in matrix-factorization-based recommender systems (Koren, Bell, and Volinsky 2009; Cremonesi, Koren, and Turrin 2010; Liu et al. 2015), multi-class prediction (Dean et al. 2013; Jain and Kapoor 2009), structural SVM (Joachims 2006; Joachims, Finley, and Yu 2009), and vision applications (Dean et al. 2013). The MIPS problem can be formally defined as follows: given a collection of n data vectors, $\mathcal{S} = \{v_1, v_2, \cdots, v_n\}$, where $v_i \in \mathbb{R}^N, 1 \leq i \leq n$, and a query vector $q \in \mathbb{R}^N$, the goal is to find $v^* \in \mathcal{S}$ that maximizes (or approximately maximizes) the inner product $q^T v^*$. In other words, MIPS is the following problem:

$$v^* = arg\max_{v \in \mathcal{S}} q^T v \tag{1}$$

The naïve linear search for solving MIPS requires $O(n \cdot N)$ time to exhaustively compute all n inner products, which can be daunting for massive datasets (large n) and/or high-dimensional data (large N). This has led to significant interest in devising sub-linear time algorithms to solve the MIPS problem approximately, including the best paper award at NIPS'14 (Shrivastava and Li 2014) and many other proposals (Yu et al. 2017; Neyshabur and Srebro 2015; Bachrach et al. 2014).

Motivation I: Preprocessing Overhead. Despite their different merits, existing approximate techniques for MIPS all share a common pattern: they require a substantial time for preprocessing the data set $\mathcal{S}$, during which they construct a data structure that can then be used to answer queries more efficiently. For example, they construct a hash table (Shrivastava and Li 2014), a sorted index (Yu et al. 2017), space partition trees (Bachrach et al. 2014), or other data structures (Auvolat et al. 2015). Existing methods require various data structures to be built during preprocessing time. In fact, the preprocessing time is, in some cases, so large that it even exceeds the time complexity of the naïve approach for answering $\log n$ queries, i.e., $O(Nn \log n)$ (Yu et al. 2017). We have summarized the preprocessing time of the previous techniques in Table 1. The rationale is that query times will be faster after the preprocessing step. The preprocessing time is therefore justified when there are many queries and the set $\mathcal{S}$ remains the same, i.e., once the preprocessing is done it would benefit many subsequent queries. However, there are many cases where either the number of queries is relatively small (or even 1), or the set $\mathcal{S}$ changes frequently, e.g., the approximate Linear Minimization Oracle (LMO) in Matching Pursuit and Frank-Wolfe optimization (Locatello et al. 2017; Jaggi 2013). In these scenarios, the preprocessing time is simply a burden. In this paper, our first motivation is to design an approach to MIPS that will not require any preprocessing, while still achieving a query speed-up better than those approaches that do require it.

Motivation II: Suboptimality Bounds. As summarized in Table 1, specific parameters such as the number of hash functions or depth of partition trees are used to trade search accuracy with search efficiency. Various data structures with pre-specified parameters (Shrivastava and Li 2014; Bachrach et al. 2014; Auvolat et al. 2015) are built before any query is given, which means that the trade-off is somewhat fixed for all queries. Thus, the computational cost for a given query is fixed. However, in many real-world scenarios, each query might have a different computational budget. (Yu et al. 2017) proposed a greedy method where the user could control the computational budget for each query.

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The computational budget can be viewed as an efficiency-accuracy knob. Nevertheless, the user can't get a solution with a guaranteed level of optimality in general.[1] It is crucial for practitioners to have a knob with which the user can explicitly request and guarantee a certain level of optimality for each query (Mozafari and Niu 2015; Agarwal et al. 2013; 2014), e.g., to know how accurate the solution would be if more computational budget were allowed (Park et al. 2018). Thus, our second motivation in this paper is to design a MIPS algorithm that can bound (and directly control) the suboptimality of the returned answer, regardless of the data distribution. Specifically, for any $0 < \epsilon < 1$ and $0 < \delta < 1$ chosen by the user, the algorithm must be able to guarantee that, with probability at least $1 - \delta$, the returned solution $\hat{v}$ is ϵ-optimal with respect to optimal solution v^*, i.e., $\frac{1}{N}q^T v^* - \frac{1}{N}q^T \hat{v} < \epsilon$. This would help us to better understand the behavior of our algorithm theoretically, and would offer a flexible knob for trading off error and computational efficiency in practice.

Our Approach. Our approach to MIPS is inspired by the Multi-Armed Bandit (MAB) problem (Bubeck, Munos, and Stoltz 2009; Audibert and Bubeck 2010; Even-Dar, Mannor, and Mansour 2006; Jamieson et al. 2014). MAB is a predominant model for characterizing the tradeoff between exploration and exploitation in decision-making settings. In MAB, there are n arms; each time we pull an arm, it returns a reward (e.g. a reward generated by sampling from a Guassian distribution). The true mean of an arm is defined as the mean of the distribution from which its rewards are sampled. The goal in MAB is to either (1) accumulate as much reward as possible, or (2) identify the best arm (i.e., the one with the highest true mean). In our paper, our goal is the latter. We cast MIPS as a Best Arm Identification problem: we can treat each data vector as an arm, where pulling it means multiplying one of its coordinates with the corresponding coordinate from the query vector. We must then dynamically decide how many more floating-point multiplications to perform for each inner product, based on the partial results of all inner products thus far.

There are two different stopping conditions for the Best Arm Identification problem: fixed confidence and fixed budget. With fixed confidence, the MAB algorithm seeks to minimize the sample complexity—the number of pulls used—while guaranteeing that the returned arm is ϵ-optimal with probability at least $1-\delta$ for any given $0 < \delta < 1, 0 < \epsilon < 1$. In the fixed budget setting, the MAB algorithm stops once it has used its budget in terms of the sample complexity, while seeking to return an arm whose true mean is as close as possible to that of the best arm. Although our **Motivation II** is inline with the fixed confidence setting, the existing MAB methods for fixed confidence are not effective for the MIPS problem. This is because the existing algorithms are designed for i.i.d. rewards drawn from some unknown distribution over an *infinite* population (Bubeck, Munos, and Stoltz 2009; Audibert and Bubeck 2010), and thus require many pulls to achieve an accurate estimate of an arm's true mean. In MIPS, however, the useful number of pulls for any arm is upper bounded by N (the vectors's dimension). In other words, rewards are sampled *without replacement* from a discrete uniform distribution over a *finite* population. Therefore, by exploiting this structure, we should be able to significantly lower the number of pulls. In this paper, we introduce a new setting for Best Arm Identification problem with fixed confidence that suits the special structure of MIPS. We also propose an algorithm, called BOUNDEDME, inspired by the Median Elimination framework (Even-Dar, Mannor, and Mansour 2002), using a tight statistical bound for sampling without replacement.

In summary, we make the following contributions:

- We identify two desirable motivations for the MIPS problem that are important in practice but overlooked by existing solutions (**Movitaion I** and **Motivation II**). We introduce a new MAB setting for Best Arm Identification, where the rewards for each arm are sampled from a large but finite list. We call this setting Multi-Arm Bandit with Bounded Pulls (MAB-BP).
- We propose a new algorithm for MAB-BP, called BOUNDEDME, which extends Median Elimination with a tight statistical bound. When applied to MIPS, BOUNDEDME enjoys a significantly lower sample complexity than all previous MAB methods developed for fixed confidence setting. More importantly, as a bandit approach, BOUNDEDME does not require any preprocessing (unlike previous MIPS solutions).
- Our extensive experiments on both synthetic and real-world datasets show that BOUNDEDME's query time is 5–10$\times$ faster than state-of-the-art MIPS algorithms, despite their use of preprocessing.

Related Work

Existing Approaches to MIPS

There are a number of sampling-based methods for MIPS. For example, SAMPLE-MIPS (Cohen and Lewis 1999) is a scheme that samples $(i, j) \in \{1, \cdots, n\} \times \{1, \cdots, N\}$ with probability proportional to $v_i^{(j)} q^{(j)}$. However, it requires that all candidate and query vectors be nonnegative. DIAMOND-MSIPS (Ballard et al. 2015) is another sampling-based approach which solves a similar problem, called *maximum squared inner product search (MSIPS)*. The goal in MSIPS is to find candidate vectors $v \in \mathcal{S}$ for which $(q^T v)^2$ is maximized. The solution to MSIPS, however, can be very different than that of MIPS, e.g., the former might return a v whose inner product with q is a large negative value.

Another popular approach is to reduce MIPS to the nearest neighbor search problem, which can then be solved using locality-sensitive hashing (Shrivastava and Li 2014; Neyshabur and Srebro 2015), neighbor-sensitive hashing (Park, Cafarella, and Mozafari 2015), PCA-trees (Bachrach et al. 2014), or K-Means approaches (Auvolat et al. 2015). Nonetheless, all of these methods share a common pattern. Before answering any queries, they conduct a preprocessing on $\mathcal{S}$ to construct an approach-specific data structure, e.g., hash table in LSH-MIPS, space partition

[1] A more detailed treatment of the previous work is deferred to Section of Related Work

Table 1: State of the art algorithms for MIPS. Here, n is the number of data vectors and N is the vectors' dimensionality.

Method	Preprocessing Time	Query Time	Theoretical Guarantees	Notes
BOUNDEDME (our method)	0	$O\left(\frac{n\sqrt{N}}{\epsilon}\sqrt{\log\left(\frac{1}{\delta}\right)}\right)$	Guaranteed to return an ϵ-optimal solution with probability at least $1-\delta$	User can choose any desired error $0<\epsilon<1$ and confidence $0<\delta<1$
GREEDY-MIPS (Yu et al. 2017)	$O(Nn\log n)$	$O(BN)$	No guarantees in general. (They guarantee optimality with high probability, only for uniformly distributed data and budget $B \geq O(N\log(n)n^{\frac{1}{N}})$	For non-uniform data, the results can be arbitrarily poor (e.g., if the largest coordinate of q^Tv is identical for all $v \in \mathcal{S}$, the output will be a random subset)
LSH-MIPS (Shrivastava and Li 2014; Neyshabur and Srebro 2015)	$O(Nnab)$	$O(\frac{nN}{2^a}b)$	They guarantee to return the optimal vector v^* for query q with probability $1-\left(1-\left(1-\frac{\cos^{-1}(q^Tv^*)}{\pi}\right)^a\right)^b$ where a is the number of bits in each hyper LSH function and b is the number of hyper LSH functions	Since v^* is unknown *a priori*, the users cannot control the lower bound of this probability (e.g., if $q^Tv^* = -1$, this probability will always be 0 regardless of the values chosen for a and b)
RPT-MIPS (Keivani, Sinha, and Ram 2017)	$O(LNn\log n)$	$O(L\log n)$	They guarantee to return the optimal vector v^* for query q with probability upper bounded by some potential function depending on q, S and L, where L is the number of trees	Since q is unknown *a priori*, the users cannot control the lower bound of this probability
PCA-MIPS (Bachrach et al. 2014)	$O(N^2n)$	$O(\frac{nN}{2^d})$	None	d is the depth of the PCA tree

trees in PCA-MIPS, or cluster centroids in (Auvolat et al. 2015). These data structures only contain information about the vectors in $\mathcal{S}$ and are independent of any query. Then, for each query, they use an efficient procedure on their preconstructed data structure to select a set of candidate vectors (i.e., a subset of $\mathcal{S}$). They perform an exact ranking on this candidate set to return the best vector. A recent approach, Randomized Partitioning Tree (RPT-MIPS) (Keivani, Sinha, and Ram 2017), builds a partitioning tree on top of an LSH scheme to solve the MIPS problem. RPT-MIPS guarantees the exact solution with a probability that depends on the vector set S and the given query. However, RPT-MIPS cannot directly control the quality of the returned vector (i.e., suboptimality bounds in **Motivation II**). Another approach is GREEDY-MIPS (Yu et al. 2017), which builds on the same algorithmic pattern, but also provides a budget for the size of the candidate set. This parameter is the only mechanism that implicitly controls the tradeoff between precision and query time. Table 1 summarizes the theoretical guarantees offered by some of the recent MIPS algorithms.

Existing Approaches to MAB

Best Arm Identification is a popular setting in MAB that aims to identify the best arm by dynamically deciding on how many times to pull each arm. In the fixed budget setting, the main idea behind algorithms such as Successive Halving (Karnin, Koren, and Somekh 2013; Jamieson and Talwalkar 2016) and Successive Rejects (Audibert and Bubeck 2010) is to dynamically allocate the budget to the different arms in order to remove the bad arms in a round-by-round fashion until only one arm is left. Naturally, these methods tend to use up their budget in order to return an arm whose true mean is as close as possible to the optimal arm. The fixed budget setting isn't suitable for our problem in this paper. In the fixed confidence setting, algorithms share the same idea of dynamically pulling arms and removing the unpromising ones from consideration, such as Successive Elimination (Even-Dar, Mannor, and Mansour 2006), Exponential Gap Elimination (Karnin, Koren, and Somekh 2013), LUCB (Kalyanakrishnan et al. 2012), and Lil'UCB (Jamieson et al. 2014). These algorithms are inline with our **Motivation II**, as they also seek to minimize sample complexity while guaranteeing that the returned arm is within a pre-specified proximity of the optimal arm. However, they cannot be directly applied to the MIPS problem because they assume the rewards are i.i.d. samples from some unknown distribution over an *infinite population*, whereas in MIPS, the rewards are sampled without replacement from a discrete uniform distribution over a *finite* list.

A Bandit Approach to MIPS

In this section, we show how the MIPS problem can be viewed as a Best Arm Identification problem with fixed confidence. As previously mentioned, the goal in MIPS is to solve the problem:

$$v^* = arg\max_{v\in\mathcal{S}} q^Tv$$

where $\mathcal{S} = \{v_i | v_i \in \mathbb{R}^N, 1 \leq i \leq n\}$ is a collection of n data vectors and $q \in \mathbb{R}^N$ is a given query.

We cast MIPS as a bandit problem as follows. For every data vector $v_i \in \mathcal{S}$, we consider a corresponding arm a_i, whose reward has the following true mean $p_i = \frac{1}{N}\sum_{j=1}^{N} v_i^{(j)} q^{(j)} = \frac{v_i^T q}{N}$, where $v_i^{(j)}$ and $q^{(j)}$ are the j-th coordinates of v_i and q, respectively, for $1 \leq j \leq N$. When the arm a_i is pulled t times ($1 \leq t \leq N$), its rewards are generated by taking t i.i.d. samples *with* replacement from its reward list R_i, defined as the set $R_i = \{v_i^{(1)} q^{(1)}, v_i^{(2)} q^{(2)}, \cdots, v_i^{(N)} q^{(N)}\}$.

Each time the arm a_i is pulled, returning a reward corresponds to a floating-point operation in MIPS for multiplying one of the coordinates from v_i with its counterpart from q, i.e., computing $v_i^{(j)} q^{(j)}$ for some $1 \leq j \leq N$. The reward lists are initially unknown, but the more we pull an arm, the more we learn about its reward list. Our goal in MIPS is to find the arm with the highest true mean—the vector whose inner product with q is the highest—using as few floating point operations as possible.

Unfortunately, in a traditional bandit setting, even if we pull the a_i arm N times, we still do not know the exact true mean of R_i, i.e., the exact inner product of v_i and q. This is because traditional bandit problems are designed for an unknown distribution over an infinite population, and hence rely on sampling *with* replacement. However, to solve MIPS, the rewards are drawn from a finite reward list—i.e., N coordinates. Thus, if we can exploit this structure and use sampling *without* replacement, we must be able to pull the arms significantly fewer times than in a traditional bandit. Further, once we have pulled the a_i arm N times, we should know the entire content of R_i, equivalent to the exact computation of the inner product between v_i and q. Next, we formally define this new bandit setting.

Multi-Armed Bandit with Bounded Pulls (MAB-BP)

We now formally introduce a new Multi-Armed Bandit setting, which we call *Multi-Armed Bandit with Bounded Pulls (MAB-BP)*. Assume a set of n arms $A = \{a_1, \cdots, a_n\}$. Each arm a_i is associated with a reward list $R_i = \{R_i^{(1)}, R_i^{(2)}, \cdots, R_i^{(N)}\}$, where N is the size of the reward list. Here, we assume $R_i^{(j)} \in [0, 1]$, but a similar analysis applies as long as the reward value is bounded. Every time an arm a_i is pulled, a reward is returned by sampling a value *without* replacement from its reward list R_i. Denote the true mean of reward for arm a_i as $p_i = \frac{1}{N}\sum_{j=1}^{N} R_i^{(j)}$. Thus, once an arm is pulled N times, the mean of the returned rewards is exactly equal to the true mean p_i. Our goal is the same as in a traditional Best Arm Identification: to identify an ϵ-optimal arm with probability at least $1 - \delta$ using as few pulls as possible, where ϵ and δ are provided by the user. We say that an arm $\hat{a}$ is an ϵ-optimal arm if $p_{a^*} - p_{\hat{a}} < \epsilon$, where a^* is the optimal arm.

It is easy to see that by choosing $R_i^{(j)} = v_i^{(j)} q^{(j)}$ for $1 \leq j \leq N$, one can cast MIPS as a MAB-BP problem. However, note that MAB-BP can be used to solve any problem of the form:

$$arg \max_{1 \leq i \leq n} \sum_{j=1}^{N} f(i, j)$$

where f can be an arbitrary function.

For MIPS, $f(i, j) = v_i^{(j)} q^{(j)}$. However, one can also use MAB-BP to solve the Nearest Neighbor Search (NNS) problem: given a collection of vectors $\mathcal{S} = \{v_1, \cdots, v_n\}$, where $v_i \in \mathbb{R}^N, 1 \leq i \leq n$, and a query vector $q \in \mathbb{R}^N$, the goal is to find $v^* \in \mathcal{S}$ that is closest to q, i.e., $\|q - v^*\|^2 = \sum_{j=1}^{N} (q^{(j)} - v^{*(j)})^2$ is minimized. In this case, $f(i, j) = -(q^{(j)} - v_i^{(j)})^2$.

BOUNDEDME: An Algorithm for Solving MAB-BP

Existing bandit algorithms are sub-optimal for MAB-BP. The fundamental reason is bandit algorithms have to estimate the minimum number of samples needed to obtain an estimate $\hat{p}$ of the true mean p that satisfies the given error ϵ and confidence δ requirements, i.e., $\mathbb{P}[\hat{p} - p \leq \epsilon] \geq 1 - \delta$.

The efficiency of a bandit algorithm depends on how accurately it can estimate the number of required samples, based on the reward values it has observed for each arm. This goal is achieved using concentration inequalities (Boucheron, Lugosi, and Massart 2013). Since in traditional bandit the rewards are typically assumed to be sampled from a sub-Gaussian distribution over an infinite population (Jamieson and Nowak 2014), these algorithms often rely on Hoeffding's bound or the law of iterated logarithm (LIL) bound to determine the sample size for mean estimation of the reward distribution over an infinite population. However, as noted earlier, the reward values in MAB-BP are sampled without replacement and from a finite list. Thus, an algorithm that can exploit this additional information should be able to solve the MAB-BP problem more efficiently (i.e., with lower sample complexity).

Next, we derive a concentration inequality for sampling without replacement and then present our algorithm.

A Concentration Inequality for MAB-BP

We use the following corollary from (Bardenet, Maillard, and others 2015).

Corollary 1 (Corollary 2.5 in (Bardenet, Maillard, and others 2015)). *Let $\mathcal{X} = (x_1, x_2, \cdots, x_N)$ be a finite set of size $N > 1$ in $[a, b]$ with mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, and $(X_1, \cdots, X_m)$ be a list of size $m < N$ sampled without replacement from $\mathcal{X}$. Then for any $m \leq N$, and any $\delta \in [0, 1]$, it holds*

$$\mathbb{P}\left[\frac{1}{m}\sum_{t=1}^{m} X_t - \mu \leq (b - a)\sqrt{\frac{\rho_m \log(1/\delta)}{2m}}\right] \geq 1 - \delta \quad (2)$$

where ρ_m is defined as

$$\rho_m = min\left\{(1 - \frac{m-1}{N}), (1 - \frac{m}{N})(1 + 1/m)\right\} \quad (3)$$

Based on this corollary, we could get the following concentration inequality for sampling without replacement, as in the following lemma[2].

Lemma 1. *Let $\mathcal{X} = (x_1, x_2, \cdots, x_N)$ be a finite set of size $N > 1$ in $[a, b]$ with mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, and $(X_1, \cdots, X_m)$ be a list of size $m < N$ sampled without replacement from $\mathcal{X}$. Then, for any given $0 < \epsilon < 1, 0 < \delta < 1$, if*

$$m = min\{\frac{u+1}{1+\frac{u}{N}}, \frac{u+\frac{u}{N}}{1+\frac{u}{N}}\} \tag{4}$$

where $u = \frac{\log(1/\delta)}{2}\frac{(b-a)^2}{\epsilon^2}$, then we have

$$\mathbb{P}\left[\frac{1}{m}\sum_{t=1}^{m} X_t - \mu \leq \epsilon\right] \geq 1 - \delta \tag{5}$$

.

We can see that as the error bound ϵ approaches 0, the required sample size m will approach the finite set size N, but never exceed N. This matches our previous intuition that it is ineffective to pull one arm more than N times in MAB-BP. It is worth noting that a lemma similar to Lemma 1 can be derived to show that $\mathbb{P}\left[\frac{1}{m}\sum_{t=1}^{m} X_t - \mu \geq -\epsilon\right] \geq 1 - \delta$, based on a corollary similar to Corollary 1.

The BOUNDEDME Algorithm

Our proposed algorithm, BOUNDEDME, is based on the median elimination strategy, but tailored to our MAB-BP setting. The basic idea of median elimination strategy is that, given a set of arms A, we pull each of these arms for a certain number of times to update their empirical means, discard the worst half in terms of their updated empirical means thus far, and repeat until only one arm remains. In Algorithm 1, we present BOUNDEDME for the more general case of identifying the top K arms with the highest true means of rewards. (The best arm identification is a special case, where K=1.) To be more specific about the difference between Algorithm 1 and the general median elimination strategy, the number of times that we pull each remaining arm is $t_l - t_{l-1}$ for the l-th iteration, where t_l is defined in the line 7 of Algorithm 1. Since we want to identify the top K arms, at the end of the l-th iteration, we discard $\left\lceil \frac{|S_l|-K}{2} \right\rceil$ arms with least empirical means thus far where $|S_l|$ is the number of remaining arms at the begining of l-th iteration, rather than discarding the worst half. In addition, we will stop when only K arms remain.

To simplify our notation, we enumerate the arms according to their true mean, i.e., $p_1 > p_2 > \cdots > p_n$. Let $T^* = \{1, 2, \cdots, K\}$ be the set of best K arms.[3] For any set T consisting of K arms, we say that T is **ϵ-optimal** if $\tilde{p}_{T^*} - \tilde{p}_T \leq \epsilon$, where $\tilde{p}_S$ is the K-th highest true mean among the arms in S. We also define the **suboptimality** of T to be $\tilde{p}_{T^*} - \tilde{p}_T$.

Given ϵ and δ provided by the user, BOUNDEDME's goal is to identify a set of K arms that is ϵ-optimal with probability at least $1 - \delta$, using as few pulls as possible.

[2]All omitted proofs can be found in the supplementary material.

[3]We use the index i instead of a_i to simplify the notation.

Algorithm 1 BOUNDEDME Algorithm (for top-K)

1: **input:** $K \geq 1, \epsilon > 0, \delta > 0$, and a set of arms A
2: **output:** a set of K arms that is ϵ-optimal with probability $1 - \delta$
3:
4: set $S_1 = A, \epsilon_1 = \frac{\epsilon}{4}, \delta_1 = \frac{\delta}{2}, l = 1$
5: set $t_0 = 0$
6: **while** $|S_l| > K$ **do**
7: set $t_l = m\left(\frac{2}{\epsilon_l^2}\log\left(\frac{2(|S_l|-K)}{\delta_l\left(\left\lfloor\frac{|S_l|-K}{2}\right\rfloor+1\right)}\right)\right)$
8: Pull every arm $a \in S_l$ for $t_l - t_{l-1}$ times, and let $\hat{p}_a^l$ denote its empirical mean since the begining of the algorithm
9: Find the $\left\lceil\frac{|S_l|-K}{2}\right\rceil$-th value of $\hat{p}_a^l$ in ascending order, and denote it as $\bar{p}_l$
10: $S_{l+1} = S_l \setminus \{a \in S_l : \hat{p}_a^l \leq \bar{p}_l\}$ (more precisely, remove $\left\lceil\frac{|S_l|-K}{2}\right\rceil$ arms with the least empirical means thus far)
11: $\epsilon_{l+1} = \frac{3}{4}\epsilon_l, \delta_{l+1} = \frac{\delta_l}{2}, l = l + 1$
12: **end while**
13: **return** S_l

We use the following function to simplify our presentation:

$$m(u) = min\left\{\frac{u+1}{1+\frac{u}{N}}, \frac{u+\frac{u}{N}}{1+\frac{u}{N}}\right\} \tag{6}$$

We can see that $m(u) < N$ as long as $u > 0$. We have the following lemma.

Lemma 2. *For Algorithm 1, at any iteration l, we have $\mathbb{P}\left[\tilde{p}_{S_l} \leq \tilde{p}_{S_{l+1}} + \epsilon_l\right] \geq 1 - \delta_l$.*

Based on the above Lemma 2, we could get the main theoretical property of Algorithm 1.

Theorem 1. *The* BOUNDEDME *algorithm (Algorithm 1) is guaranteed to return ϵ-optimal solution with probability at least $1 - \delta$.*

Note that BOUNDEDME is never slower than the naïve search, which has the $O(nN)$ time complexity:

Corollary 2. *For each arm, the number of times it is pulled by Algorithm 1 is upper-bounded by N.*

BOUNDEDME's time complexity is also lower than Median Elimination (Even-Dar, Mannor, and Mansour 2002), which is $O\left(\frac{n}{\epsilon^2}\log(\frac{1}{\delta})\right)$:

Corollary 3. *The time complexity of Algorithm 1 is $O\left(\frac{n\sqrt{N}}{\epsilon}\sqrt{\log\left(\frac{1}{\delta}\right)}\right)$.*

Remark 1. *When Algorithm 1 is applied to the MIPS problem, the above bound indicates that the running time is sub-linear in the dimension of vectors, but linear in the size of vector set $\mathcal{S}$. This implies that our approach is especially effective for very high-dimensional data. To mitigate the potential issue of linear dependence on the size of $\mathcal{S}$, we could exploit the geometric structure or similarity among vectors from $\mathcal{S}$. The tradeoff here is that this would now require*

some preprocessing. For example, we could find the convex hull of the set $\mathcal{S}$ first, and then only focus on the set of extreme points that form the convex hull, because the solution of the MIPS problem is guaranteed to always include at least one of these extreme points. Thus, when the number of extreme points is much smaller than the size of $\mathcal{S}$, our algorithm becomes sublinear in the size of $\mathcal{S}$.

Experiments

Our experiments aim to (1) empirically validate the theoretical guarantees of Theorem 1 and (2) compare our method against the state-of-the-art.

Datasets. Since Theorem 1 is a worst-case guarantee, we use an adversarily-generated synthetic dataset to verify its correctness. Then, we use both synthetic and real-world datasets to compare our algorithm with several state-of-the-art techniques. For each dataset, we used 10^4 vectors with 10^5 dimensions.

Baselines. We compared our method against the following state-of-the-art methods:

- LSH-MIPS (Shrivastava and Li 2014; Neyshabur and Srebro 2015), which is a popular method for MIPS. We used the nearest neighbor transformation proposed in (Bachrach et al. 2014) and the LSH function, as suggested in (Neyshabur and Srebro 2015). We used the standard amplification procedure, i.e., the final result is an OR-construction of b hyper LSH hash functions and each hyper LSH function is an AND-construction of a random projections.
- GREEDY-MIPS (Yu et al. 2017), which is a recently proposed method that uses a budget B to control the time complexity of the query time.
- PCA-MIPS (Bachrach et al. 2014), which uses the depth of the PCA tree to control the time-precision tradeoff.

Comparison Metrics. We compare different algorithms by varying their parameters in order to explore their tradeoffs between precision and online speedup. Precision is defined as the fraction of true top K solutions in the returned top K solutions. Online speedup of an algorithm is defined as the query time required by the naïve (i.e., exhaustive) search divided by the query time of that algorithm. Recall that, unlike the baselines, our algorithm does not require any preprocessing. However, we ignore the preprocessing time of the baselines in our comparisons, showing that our algorithm's online speedup is still superior despite the lack of any preprocessing.

Characteristics of the BOUNDEDME Algorithm

Theorem 1 provides a PAC bound for BOUNDEDME. In other words, with the δ and ϵ provided by the user, BOUNDEDME is guaranteed to return an ϵ-optimal solution with probability at least $1-\delta$. Note that this is a worst-case guarantee, and in most cases we expect the returned solution to be much better than that. Therefore, to empirically validate our worst-case guarantee, we design an adversarial dataset as follows (we will use other realistic datasets in later experiments).

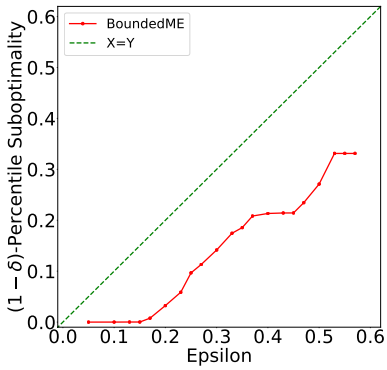
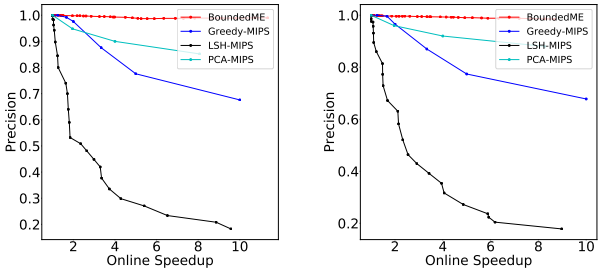


Figure 1: Correctness of BOUNDEDME's guarantees



(a) top 5 solutions (b) top 10 solutions

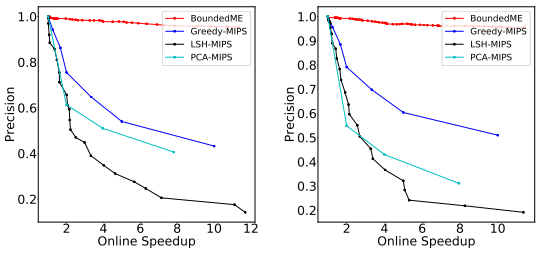
Figure 2: Synthetic Gaussian dataset

To generate an adversarial dataset, we use 10^4 arm, each with a list of 10^5 reward values. For each arm a, we choose its true mean r_a uniformly at random from $[0, 1]$. Then, the rewards for that arm are generated with every reward being 1 with probability r_a and being 0 with probability $1 - r_a$. When an arm is pulled—i.e., a sample is drawn from the rewards list without replacement—the rewards with value 1 are returned before those with value 0. This is to make the arms as indistinguishable as possible to the algorithm, thus causing an adversarial scenario.

In this experiment, we vary ϵ between 0 and 0.6. For each value of ϵ, we try all values of δ from the set $\{0.01, 0.05, 0.1, 0.2, 0.3\}$. For each pair of ϵ and δ, we run BOUNDEDME 20 times, each time on a different randomly generated adversarial dataset (as described above). We then measure the $(1-\delta)$-percentile of the list of suboptimalities for each specific pair of ϵ and δ. Figure 1 reports the average of these suboptimalities for each value of ϵ. Since the suboptimality is always less than its corresponding value of ϵ, it confirms that they are indeed smaller than their corresponding values of ϵ, i.e., validating Theorem 1.

BOUNDEDME vs. Other MIPS Algorithms on Synthetic Datasets

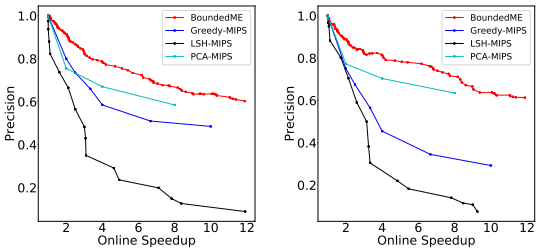
We generate two synthetic datasets, where the vector values are drawn from Gaussian and uniform distributions, respectively. For BOUNDEDME, we varied $\epsilon, \delta \in [0, 1]$. For LSH-



(a) top 5 (b) top 10

Figure 3: Synthetic uniform dataset

MIPS, we varied $a \in [1, 20]$ and $b \in [1, 50]$. For GREEDY-MIPS, we varied B from 10% to 100% of the dataset size. For PCA-MIPS, we varied the tree depth in $[0, 20]$. We run experiments for both the cases of returning the top 5 and 10 solutions. As shown in Figures 2 and 3, when the online speedup is small, all methods have very high precision. However, when online speedup becomes larger, the precisions achieved by other methods start to drop quickly, while BOUNDEDME can still maintain high precision. This demonstrates that BOUNDEDME outperforms these previous methods, despite its lack of preprocessing time.



(a) Netflix Dataset (b) Yahoo Dataset

Figure 4: Real-world datases

BOUNDEDME vs. Other MIPS Algorithms on Real-World Datasets

We also compare BOUNDEDME against others on two real-world datasets, Netflix and Yahoo-Music used in (Yu et al. 2017). We use the same setting as in (Yu et al. 2017) to compute the vector embeddings using matrix factorization. The other parameters are the same as previous subsection for the case of identifying the top 5 solutions. Again, as shown in Figure 4, the precision of other methods drops more quickly than that of BOUNDEDME, implying the superior performance of BOUNDEDME over its counterparts.

Conclusion

We introduced a new bandit setting, Multi-Armed Bandit with Bounded Pulls (MAB-BP), where the rewards are sampled *without* replacement from a *finite* list. We showed that this setting can be used for solving important problems, such as Maximum Inner Product Search and Nearest Neighbor Search. We also proposed a new algorithm, BOUNDEDME, which extends the Median Elimination framework for MAB-BP settings. Using a new concentration inequality for finite lists, we derived BOUNDEDME's suboptimality guarantee and sample complexity. By applying BOUNDEDME to MIPS, we improved on state-of-the-art methods for MIPS by (1) avoiding their preprocessing step, and (2) offering a knob to the user to directly control the suboptimaltiy of the results. We also conducted extensive experiments on both synthetic and real-world datasets, showing significant speedups over state-of-the-art MIPS algorithms.

Acknowledgments

This work is in part supported by National Science Foundation (grants 1629397 and 1553169). The authors would like to thank anynomous reviewers for their insightful comments.

References

Agarwal, S.; Mozafari, B.; Panda, A.; Milner, H.; Madden, S.; and Stoica, I. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys*.

Agarwal, S.; Milner, H.; Kleiner, A.; Talwalkar, A.; Jordan, M.; Madden, S.; Mozafari, B.; and Stoica, I. 2014. Knowing when you're wrong: Building fast and reliable approximate query processing systems. In *SIGMOD*.

Audibert, J.-Y., and Bubeck, S. 2010. Best arm identification in multi-armed bandits. In *COLT-23th Conference on Learning Theory-2010*, 13–p.

Auvolat, A.; Chandar, S.; Vincent, P.; Larochelle, H.; and Bengio, Y. 2015. Clustering is efficient for approximate maximum inner product search. *arXiv preprint arXiv:1507.05910*.

Bachrach, Y.; Finkelstein, Y.; Gilad-Bachrach, R.; Katzir, L.; Koenigstein, N.; Nice, N.; and Paquet, U. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, 257–264. ACM.

Ballard, G.; Kolda, T. G.; Pinar, A.; and Seshadhri, C. 2015. Diamond sampling for approximate maximum all-pairs dot-product (mad) search. In *Data Mining (ICDM), 2015 IEEE International Conference on*, 11–20. IEEE.

Bardenet, R.; Maillard, O.-A.; et al. 2015. Concentration inequalities for sampling without roeplacement. *Bernoulli* 21(3):1361–1385.

Boucheron, S.; Lugosi, G.; and Massart, P. 2013. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press.

Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, 23–37. Springer.

Cohen, E., and Lewis, D. D. 1999. Approximating matrix

multiplication for pattern recognition tasks. *Journal of Algorithms* 30(2):211–252.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, 39–46. ACM.

Dean, T.; Ruzon, M. A.; Segal, M.; Shlens, J.; Vijayanarasimhan, S.; and Yagnik, J. 2013. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 1814–1821. IEEE.

Even-Dar, E.; Mannor, S.; and Mansour, Y. 2002. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, 255–270. Springer.

Even-Dar, E.; Mannor, S.; and Mansour, Y. 2006. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research* 7(Jun):1079–1105.

Jaggi, M. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, 427–435.

Jain, P., and Kapoor, A. 2009. Active learning for large multi-class problems. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 762–769. IEEE.

Jamieson, K., and Nowak, R. 2014. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, 1–6. IEEE.

Jamieson, K., and Talwalkar, A. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, 240–248.

Jamieson, K.; Malloy, M.; Nowak, R.; and Bubeck, S. 2014. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, 423–439.

Joachims, T.; Finley, T.; and Yu, C.-N. J. 2009. Cutting-plane training of structural svms. *Machine Learning* 77(1):27–59.

Joachims, T. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 217–226. ACM.

Kalyanakrishnan, S.; Tewari, A.; Auer, P.; and Stone, P. 2012. Pac subset selection in stochastic multi-armed bandits. In *ICML*, volume 12, 655–662.

Karnin, Z.; Koren, T.; and Somekh, O. 2013. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, 1238–1246.

Keivani, O.; Sinha, K.; and Ram, P. 2017. Improved maximum inner product search with better theoretical guarantees. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, 2927–2934. IEEE.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8).

Liu, R.; Cheng, W.; Tong, H.; Wang, W.; and Zhang, X. 2015. Robust multi-network clustering via joint cross-domain cluster alignment. In *Data Mining (ICDM), 2015 IEEE International Conference on*, 291–300. IEEE.

Locatello, F.; Khanna, R.; Tschannen, M.; and Jaggi, M. 2017. A unified optimization view on generalized matching pursuit and frank-wolfe. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-229229.

Mozafari, B., and Niu, N. 2015. A handbook for building an approximate query engine. *IEEE Data Eng. Bull.*

Neyshabur, B., and Srebro, N. 2015. On symmetric and asymmetric LSHs for inner product search. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, 1926–1934. JMLR. org.

Park, Y.; Qing, J.; Shen, X.; and Mozafari, B. 2018. BlinkML: Approximate machine learning with probabilistic guarantees y. *Technical Report* http://web.eecs.umich.edu/~mozafari/php/data/uploads/blinkml_report.pdf.

Park, Y.; Cafarella, M.; and Mozafari, B. 2015. Neighbor-sensitive hashing. *PVLDB*.

Shrivastava, A., and Li, P. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, 2321–2329.

Yu, H.-F.; Hsieh, C.-J.; Lei, Q.; and Dhillon, I. S. 2017. A greedy approach for budgeted maximum inner product search. In *Advances in Neural Information Processing Systems*, 5459–5468.