Faster Maximum Inner Product Search in High Dimensions

Anonymous submission

Abstract

Maximum Inner Product Search (MIPS) is a ubiquitous task in machine learning applications such as recommendation systems. Given a query vector and n atom vectors in d-dimensional space, the goal of MIPS is to find the atom that has the highest inner product with the query vector. Existing MIPS algorithms scale at least as $O(\sqrt{d})$, which becomes computationally prohibitive in high-dimensional settings that are prevalent in various real-world scenarios. In this work, we present BanditMIPS, a novel randomized MIPS algorithm whose complexity is independent of d. BanditMIPS estimates the inner product for each atom by adaptively subsampling coordinates for more promising atoms, a strategy motivated by multi-armed bandits. We provide theoretical guarantees that BanditMIPS returns the correct answer with high probability, while improving the complexity in d from $O(\sqrt{d})$ to $O(1)$. We also perform experiments on four synthetic and real-world datasets and demonstrate that BanditMIPS outperforms prior state-of-the-art algorithms. For example, in the Movie Lens dataset (n=4,000, d=6,000), BanditMIPS is 20$\times$ faster than the next best algorithm while returning the same answer. BanditMIPS requires no preprocessing of the data and includes a hyperparameter that practitioners may use to trade off accuracy and runtime. We also propose a variant of our algorithm, named BanditMIPS-α, which achieves further speedups by employing non-uniform sampling across coordinates, and demonstrate how known preprocessing techniques can be used to further accelerate BanditMIPS. Finally, we illustrate the potential of BanditMIPS as a versatile subroutine, enabling any machine learning algorithms that employ MIPS (e.g. Matching Pursuit, Hierarchical Navigable Small Worlds, and a classification layer in a large language model) to harness rich high-dimensional datasets without the need for dimensionality reduction.

1 Introduction

The Maximum Inner Product Search problem (MIPS) (Shrivastava and Li 2014a; Neyshabur and Srebro 2015; Yu et al. 2017) is a ubiquitous task that arises in many machine learning applications, such as matrix-factorization-based recommendation systems (Koren, Bell, and Volinsky 2009; Cremonesi, Koren, and Turrin 2010), multi-class prediction (Dean et al. 2013; Jain and Kapoor 2009), structural SVM (Joachims 2006; Joachims, Finley, and Yu 2009), and computer vision (Dean et al. 2013). Given a *query* vector $\mathbf{q} \in \mathbb{R}^d$ and n *atom* vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$, MIPS aims to find the atom most similar to the query:

$$i^* = \underset{i \in \{1, \cdots, n\}}{\arg\max}\ \mathbf{v}_i^T \mathbf{q} \tag{1}$$

For example, in recommendation systems, the query $\mathbf{q}$ may represent a user and the atoms $\mathbf{v}_i$'s represent items with which the user can interact; MIPS finds the best item for the user, as modeled by their concordance $\mathbf{v}_i^T \mathbf{q}$ (Amagata and Hara 2021; Aouali et al. 2022). In many applications, the number of atoms n and the feature dimension d can easily be in the millions, so it is critical to solve MIPS accurately and efficiently (Hirata et al. 2022).

The naïve approach evaluates all nd elements and scales as $O(nd)$. Prior methodologies have aimed to reduce scaling with n by reconstructing the underlying data structure which requires heavy preprocessing, especially on datasets with large d (Morozov and Babenko 2018a; Liu et al. 2020). To avoid this overhead, more recent works such as (Lorenzen and Pham 2021) and (Liu, Wu, and Mozafari 2019) advocate for sampling-based approaches. However, the complexity remains at best $O(\sqrt{d})$, which is not ideal, considering the prevalence of high dimensional datasets in domains such as e-commerce, genomics, and finance. Current practices attempt to address this issue with dimensionality reduction techniques, but this induces information loss (particularly in higher dimensions) and tends to scale with $O(d)$(Li and Wan 2020).

To this end, we propose BanditMIPS, a state-of-the-art randomized algorithm that solves MIPS problems on the fly. We demonstrate BanditMIPS 's dimensionality-independent complexity and provide a tunable hyperparameter that governs the tradeoff between accuracy and speed, a need identified by previous works (Yu et al. 2017). We also provide theoretical guarantees that BanditMIPS recovers the exact solution to Equation (1) with high probability in $\tilde{O}(\frac{n}{\Delta^2})^*$ time, where Δ is an instance-specific factor that does not depend on d. We have also performed comprehensive experiments to evaluate our algorithm's performance in two synthetic and two real-world datasets. For example, in the Movie Lens dataset ($n = 4,000$, $d = 6,000$) (Harper and Konstan 2015), BanditMIPS is 20$\times$ faster than prior state-of-the-art while returning the same answer.

*The $\tilde{O}$ notation hides logarithmic factors.

At a high-level, instead of computing the inner product $\mathbf{v}_i^T\mathbf{q}$ for each atom i using all d coordinates, BanditMIPS estimates them by subsampling a subset of coordinates. Since more samples give higher estimation accuracy, BanditMIPS adaptively samples more coordinates for top atoms to discern the best atom. The specific adaptive sampling procedure is motivated by multi-armed bandits (MAB) (Even-Dar, Mannor, and Mansour 2006).

BanditMIPS is easily parallelizable and can be used with other optimization objectives that decompose coordinate-wise. Unlike previous works, it does not require preprocessing, dimensionality reduction, or normalization of the data, nor does it require the query or atoms to be nonnegative (Yu et al. 2017). This versatility allows BanditMIPS to work as a flexible subroutine for various machine learning tasks and also lends itself to specific extensions as shown below. In summary, our work offers the following contributions:

- **Novel Algorithm: BanditMIPS .** We introduce BanditMIPS , a new state-of-the-art algorithm for MIPS in high-dimensional settings. Achieving $O(1)$ sample complexity with respect to dimensionality eliminates the need for preprocessing and dimensionality reduction techniques, empowering rich high-dimensional data processing.
- **Three Algorithmic Extensions.** First, we propose BanditMIPS-α, which provides additional runtime speedups by sampling coordinates intelligently (Section 3). Second, we extend BanditMIPS to find the k atoms with the highest inner products with the query (k-MIPS) in our experiments (Section 5). Third, we discuss how BanditMIPS can be used in conjunction with preprocessing techniques leading to complexity reductions in both the number n and dimension d of the dataset (Appendix 5).
- **Versatile Integration**. We accelerate machine learning applications such as Orthogonal Matching Pursuit and a classification layer of a large language model using BanditMIPS as a black-box subroutine (Appendix 10).
- **Empirical Superiority**. We demonstrate that BanditMIPS outperforms rivals, achieving up to $30\times$ efficiency over the next best algorithm due to reduced sample usage in real datasets.

Related work

MIPS applications: MIPS arises naturally in many information retrieval contexts (Sivic and Zisserman 2003; Dong et al. 2012; Boytsov et al. 2016) and for augmenting large, auto-regressive language models (Borgeaud et al. 2022). MIPS is also a subroutine in the Matching Pursuit problem (MP) and its variants, such as Orthogonal Matching Pursuit (OMP) (Locatello et al. 2017). MP and other iterative MIPS algorithms have found many applications, e.g., to find a sparse solution of underdetermined systems of equations (Donoho et al. 2012) and accelerate conditional gradient methods (Song et al. 2022; Xu, Song, and Shrivastava 2021). MIPS also arises in the inference stages of many other applications, such as for deep-learning based multi-class or multi-label classifiers (Dean et al. 2013; Jain and Kapoor 2009) and has been used as a black-box subroutine to improve the learning and inference in unnormalized log-linear models when computing the partition function is intractable (Mussmann and Ermon 2016).

MIPS algorithms: Many approaches focus on solving approximate versions of MIPS. Such work often assumes that the vector entries are nonnegative, performs non-adaptive sampling (Lu, Hu, and Zeng 2017; Ballard et al. 2015; Lorenzen and Pham 2021; Ding, Yu, and Hsieh 2019; Yu et al. 2017), or rely on product quantization (Dai et al. 2020; Wu et al. 2019; Guo et al. 2020, 2019; Matsui et al. 2018; Douze, Jégou, and Perronnin 2016; Ge et al. 2013; Babenko and Lempitsky 2012; Jégou et al. 2011; Jégou, Douze, and Schmid 2011). Many of these algorithms require significant preprocessing, are limited in their adaptivity to the underlying data distribution, provide no theoretical guarantees, or scale linearly in d—all drawbacks that have been identified as bottlenecks for MIPS in high dimensions (Ponomarenko et al. 2014).

A large family of MIPS algorithms are based on locality-sensitive hashing (LSH) (Indyk and Motwani 1998; Shrivastava and Li 2014a, 2015; Neyshabur and Srebro 2015; Huang et al. 2015; Song et al. 2021; Lu and Kudo 2021; Shrivastava and Li 2014b; Wu et al. 2022; Huang et al. 2018; Ma et al. 2021; Andoni et al. 2015; Yan et al. 2018). A shortcoming of these LSH-based approaches is that, in high dimensions, the maximum dot product is often small compared to the vector norms, which necessitates many hashes and significant storage space (often orders of magnitude more than the data itself). Many other MIPS approaches are based on proximity graphs, such as ip-NSW (Morozov and Babenko 2018a) and related work (Liu et al. 2020; Feng et al. 2023; Tan et al. 2019, 2021; Zhou et al. 2019; Chen et al. 2022; Zhang, Wang, and He 2022; Alexander et al. 2011; Malkov and Yashunin 2016; Malkov et al. 2014). These approaches use preprocessing to build an index data structure that allows for more efficient MIPS solutions at query time. However, these approaches also do not scale well to high dimensions as the index structure (an approximation to the true proximity graph) is subject to the curse of dimensionality (Liu et al. 2020).

Perhaps most similar to our work is BoundedME which solves the MIPS problem based on an adaptive sampling approach (Liu, Wu, and Mozafari 2019). However, this method scales as $O(n\sqrt{d})$ which is objectively inferior to BanditMIPS 's independence with dimension d. The worse scaling comes from predetermining the number of times each atom is sampled by d and not adapting to the actual *values* of the sampled inner products as done in BanditMIPS; rather, BoundedME is only adaptive to the relative *ranking* of the inner products. Intuitively, this approach is wasteful because information contained in the sampled inner product's values is discarded. Additional related work is discussed in Appendix 7.

Multi-armed bandits: BanditMIPS is motivated by the best-arm identification problem in multi-armed bandits (Even-Dar, Mannor, and Mansour 2006; Karnin, Koren, and Somekh 2013; Audibert, Bubeck, and Munos 2010; Jamieson and Nowak 2014; Jamieson et al. 2014; Jamieson and Talwalkar 2016; Bubeck, Munos, and Stoltz 2011; Bardenet and Mail-

lard 2015; Boucheron, Lugosi, and Massart 2013; Even-Dar, Mannor, and Mansour 2002; Kalyanakrishnan et al. 2012). In a typical setting, we have n arms each associated with an expected reward μ_i. At each time step $t = 0, 1, \cdots$, we decide to pull an arm $A_t \in \{1, \cdots, n\}$, and receive a reward X_t with $E[X_t] = \mu_{A_t}$. The goal is to identify the arm with the largest reward with high probability while using the fewest number of arm pulls. The use of MAB-based adaptive sampling to develop computationally efficient algorithms has seen many applications, such as random forests and k-medoid clustering (Tiwari et al. 2020; Bagaria et al. 2018; Bagaria, Kamath, and Tse 2018; Zhang, Zou, and Tse 2019a; Bagaria et al. 2021).

2 Preliminaries and Notation

We consider a query $\mathbf{q} \in \mathbb{R}^d$ and n atoms $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$. Let $[n]$ denote $\{1, \ldots, n\}$, q_j the jth element of $\mathbf{q}$, and v_{ij} the jth element of $\mathbf{v}_i$. For a given query $\mathbf{q} \in \mathbb{R}^d$, the MIPS problem is to find the solution to Equation (1): $i^* = \arg\max_{i \in [n]} \mathbf{v}_i^T \mathbf{q}$.

We let $\mu_i := \frac{\mathbf{v_i}^T \mathbf{q}}{d}$ denote the *normalized inner product* for atom $\mathbf{v}_i$. Since the inner products $\mathbf{v_i}^T \mathbf{q}$ tend to scale linearly with d (e.g., if each coordinate of the atoms and query are drawn i.i.d.), each μ_i should not scale with d. Note that $\arg\max_{i \in [n]} \mathbf{v}_i^T \mathbf{q} = \arg\max_{i \in [n]} \mu_i$ so it is sufficient to find the atom with the highest μ_i. Furthermore, for $i \neq i^*$ we define the gap of atom i as $\Delta_i := \mu_{i^*} - \mu_i \geq 0$ and the minimum gap as $\Delta := \min_{i \neq i^*} \Delta_i$. We primarily focus on the computational complexity of MIPS with respect to d.

3 Algorithm

Algorithm 1 BanditMIPS

Input: Atoms $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$, query $\mathbf{q} \in \mathbb{R}^d$, error probability δ, sub-Gaussian parameter σ **Output**: $i^* = \arg\max_{i \in [n]} \mathbf{q}^T \mathbf{v_i}$

$\mathcal{S}_{\text{solution}} \leftarrow [n]$
$d_{\text{used}} \leftarrow 0$
For all $i \in \mathcal{S}_{\text{solution}}$, initialize $\hat{\mu}_i \leftarrow 0$, $C_{d_{\text{used}}} \leftarrow \infty$
while $d_{\text{used}} < d$ and $|\mathcal{S}_{\text{solution}}| > 1$ **do**
    Sample a new coordinate $J \sim \text{Unif}[d]$
    for all $i \in \mathcal{S}_{\text{solution}}$ **do**
        $\hat{\mu}_i \leftarrow \frac{d_{\text{used}} \hat{\mu}_i + v_{iJ} q_J}{d_{\text{used}}+1}$
        $\left(1 - \frac{\delta}{2nd_{\text{used}}^2}\right)$-CI: $C_{d_{\text{used}}} \leftarrow \sigma \sqrt{\frac{2 \log\left(4nd_{\text{used}}^2/\delta\right)}{d_{\text{used}}+1}}$
    $\mathcal{S}_{\text{solution}} \leftarrow \{i : \hat{\mu}_i + C_{d_{\text{used}}} \geq \max_{i'} \hat{\mu}_{i'} - C_{d_{\text{used}}}\}$
    $d_{\text{used}} \leftarrow d_{\text{used}} + 1$
If $|\mathcal{S}_{\text{solution}}| > 1$, update $\hat{\mu}_i$ to be the exact value $\mu_i = \mathbf{v}_i^T q$ for each atom in $\mathcal{S}_{\text{solution}}$ using all d coordinates
return $i^* = \arg\max_{i \in \mathcal{S}_{\text{solution}}} \hat{\mu}_i$

The BanditMIPS algorithm is described in Algorithm 1 and is motivated by best-arm identification algorithms. As summarized in Table 1, we can view each atom $\mathbf{v}_i$ as an arm with the arm parameter $\mu_i := \frac{\mathbf{v_i}^T \mathbf{q}}{d}$. When pulling an arm i, we randomly sample a coordinate $J \sim \text{Unif}[d]$ and evaluate the inner product at the coordinate as $X_i = q_J v_{iJ}$. Using this reformulation, the best atom can be estimated using techniques from best-arm algorithms.

BanditMIPS can be viewed as a combination of UCB and successive elimination (Lai and Robbins 1985; Even-Dar, Mannor, and Mansour 2006; Zhang, Zou, and Tse 2019b). Algorithm 1 uses the set $\mathcal{S}_{\text{solution}}$ to track all potential solutions to Equation (1); $\mathcal{S}_{\text{solution}}$ is initialized as the set of all atoms $[n]$. We will assume that, for a fixed atom i and a randomly sampled coordinate, the random variable $X_i = q_J v_{iJ}$ is σ-sub-Gaussian for some known parameter σ. With this assumption, Algorithm 1 maintains a mean objective estimate $\hat{\mu}_i$ and confidence interval (CI) for each potential solution $i \in \mathcal{S}_{\text{solution}}$, where the CI depends on the error probability δ as well as the sub-Gaussian parameter σ. We discuss the sub-Gaussian parameter and possible relaxations of this assumption in Subsections 3 and 4.

Additional speedup techniques

Non-uniform sampling reduces variance: In the original version of BanditMIPS, we sample a coordinate J for all atoms in $\mathcal{S}_{\text{solution}}$ uniformly from the set of all coordinates $[d]$. However, some coordinates may be more informative of the inner product than others. For example, larger entries of $\mathbf{v}_i$ may contribute more to the inner product with $\mathbf{q}$. As such, we sample each coordinate $j \in [d]$ with probability $w_j \propto q_j^{2\beta}$ and $\sum_j w_j = 1$, and estimate the arm parameter μ_i of atom i as $X = \frac{1}{w_J} q_J v_{iJ}$. X is an unbiased estimator of μ_i and the specific choice of coordinate sampling weights minimizes the combined variance of X across all atoms; different values of β corresponds to the minimizer under different assumptions. We provide theoretical justification of this weighting scheme in Section 4. We note that the effect of this non-uniform sampling will only accelerate the algorithm.

Warm start increases speed: One may wish to perform MIPS for a batch of m queries instead of just a single query, solving m separate MIPS problems. In this case, we can cache the atom values for all atoms across a random subset of coordinates, and provide a warm start to BanditMIPS by using these cached values to update arm parameter estimates $\hat{\mu}_i$, C_i, and $\mathcal{S}_{\text{solution}}$ for all m MIPS problems. Such a procedure will eliminate the obviously less promising atoms and avoid repeated sampling for each of the m MIPS problems and increases computational efficiency. We note that, since the m MIPS problems are independent, the theoretical guarantees described in Section 4 still hold across all m MIPS problems simultaneously.

Sub-Gaussian assumption and construction of confidence intervals

Crucial to the accuracy of Algorithm 1 is the construction of the $(1-\delta)$-CI based on the σ-sub-Gaussianity of each $X_i = q_J v_{iJ}$. We note that the requirement for σ-sub-Gaussianity is rather general. In particular, when the coordinate-wise products between the atoms and query are bounded in $[a, b]$, then each X_i is $\frac{b^2-a^2}{4}$-sub-Gaussian. This is commonly the case, e.g., in recommendation systems where user ratings

Table 1: MIPS as a best-arm identification problem.

Terminology	Best-arm identification	MIPS
Arms	$i = 1, \ldots, n$	Atoms $\mathbf{v}_1, \ldots, \mathbf{v}_n$
Arm parameter μ_i	Expected reward $\mathbb{E}[X_i]$	Average coordinate-wise product $\frac{\mathbf{v}_i{}^T \mathbf{q}}{d}$
Pulling arm i	Sample a reward X_i	Sample a coordinate J with reward $q_J v_{iJ}$
Goal	Identify best arm with probability at least $1-\delta$	Identify best atom with probability at least $1-\delta$

(each element of the query and atoms) are integers between 0 and 5, and we use this implied value of σ in our experiments in Section 5.

The $\frac{b^2-a^2}{4}$-sub-Gaussianity assumption allows us to compute $1-\delta$ CIs via Hoeffding's inequality, which states that for any random variable $S_n = Y_1 + Y_2 + \ldots Y_n$ where each $Y_i \in [a, b]$

$$P(|S_n - \mathbb{E}[S_n]| > \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{n(b-a)^2}\right).$$

Setting δ equal to the right hand side and solving for ϵ gives the width of the confidence interval. $\sigma = \frac{b^2-a^2}{4}$ acts as a variance proxy used in the creation of the confidence intervals by BanditMIPS; smaller variance proxies should result in tighter confidence intervals and lower sample complexities and runtimes.

In other settings where the sub-Gaussianity parameter may not be known *a priori*, it can be estimated from the data or the CIs can be constructed using the empirical Bernstein inequality instead (Maurer and Pontil 2009).

4 Theoretical Analysis

Analysis of the Algorithm: For Theorem 1, we assume that, for a fixed atom $\mathbf{v}_i$ and d_{used} randomly sampled coordinates, the $(1-\delta')$ confidence interval scales as $C_{d_{\text{used}}}(\delta') = O\left(\sqrt{\frac{\log 1/\delta'}{d_{\text{used}}}}\right)$ (note that we use d_{used} and δ' here because we have already used d and δ). We note that the sub-Gaussian CIs satisfy this property, as described in Section 3.

Theorem 1. *Assume* $\exists\ c_0 > 0$ *s.t.* $\forall\ \delta' > 0$, $d_{used} > 0$, $C_{d_{used}}(\delta') < c_0\sqrt{\frac{\log 1/\delta'}{d_{used}}}$. *With probability at least* $1-\delta$, *BanditMIPS returns the correct solution to Equation* (1) *and uses a total of* M *computations, where*

$$M \leq \sum_{i \in [n], i \neq i^*} \min\left[\frac{16c_0^2}{\Delta_i^2}\log\left(\frac{n}{\delta\Delta_i}\right) + 1, 2d\right]. \quad (2)$$

Theorem 1 is proven in the appendices. We note that c_0 is the sub-Gaussianity parameter described in Section 3 and is a constant. Intuitively, Theorem 1 states that with high probability, BanditMIPS returns the atom with the highest inner product with $\mathbf{q}$. The instance-wise bound Equation (2) suggests the computational cost of a given atom $\mathbf{v}_i$, i.e., $\min\left[\frac{16c_0^2}{\Delta_i^2}\log\left(\frac{n}{\delta\Delta_i}\right) + 1, 2d\right]$, depends on Δ_i, which measures how close its optimization parameter μ_i is to μ_{i^*}. Most reasonably different atoms $i \neq i^*$ will have a large Δ_i and incur an $O\left(\frac{1}{\Delta^2}\log\frac{n}{\delta\Delta_i}\right)$ computation that is independent of d when d is sufficiently large.

Important to Theorem 1 is the assumption that we can construct $(1-\delta')$ CIs $C_i(d_{\text{used}}, \delta')$ that scale as $O(\sqrt{\frac{\log 1/\delta'}{d_{\text{used}}}})$. As discussed in Section 3, this is under general assumptions, for example when the estimator $X_i = q_J v_{iJ}$ for each arm parameter μ_i has finite first and second moments (Catoni 2012) or is bounded.

Since each coordinate-wise multiplication only incurs $O(1)$ computational overhead to update running means and confidence intervals, sample complexity bounds translate directly to wall-clock times bounds up to constant factors. For this reason, our approach of focuses on sample complexity bounds, in line with prior work (Tiwari et al. 2020; Bagaria, Kamath, and Tse 2018).

Discussion of the hyperparameter δ: The hyperparameter δ allows users to trade off accuracy and runtime when calling Algorithm 1. A smaller value of δ corresponds to a lower error probability, but will lead to longer runtimes because the confidence intervals constructed by Algorithm 1 will be wider and atoms will be filtered more slowly. Theorem 1 provides an analysis of the effect of δ and in Section 5, we discuss appropriate ways to tune it. We note that setting $\delta = 0$ reduces Algorithm 1 to the naïve algorithm for MIPS. In particular, Algorithm 1 is never worse in big-O sample complexity than the naïve algorithm.

Discussion of the importance of Δ: In general, BanditMIPS takes only $O\left(\frac{1}{\Delta^2}\log\frac{n}{\delta\Delta}\right)$ computations per atom if there is reasonable heterogeneity among them. As proven in Appendix 2 in (Bagaria et al. 2018), this is the case under a wide range of distributional assumptions on the μ_i's, e.g., when the μ_i's follow a sub-Gaussian distribution across the atoms. These assumptions ensure that BanditMIPS has an overall complexity of $O\left(\frac{n}{\Delta^2}\log\frac{n}{\delta\Delta}\right)$ that is independent of d when d is sufficiently large and Δ does not depend on d.

At first glance, the assumption that each Δ_i (and therefore Δ) does not depend on d may seem restrictive. However, such an assumption actually applies under a reasonable number of data-generating models. For example, if the atoms' coordinates are drawn from a latent variable model, i.e., the μ_i's are fixed in advance and the atoms' coordinates correspond to instantiations of a random variable with mean μ_i, then Δ_i will be independent of d. As a concrete example, two users' $0/1$ ratings of movies may agree on 60% of movies and their atoms' coordinates correspond to observations of a

Bernoulli random variable with parameter 0.6. Other recent works provide further discussion on the conversion between an instance-wise bound like Equation (2) and an instance-independent bound that is independent of d (Bagaria et al. 2018; Baharav and Tse 2019; Tiwari et al. 2020; Bagaria et al. 2021; Baharav et al. 2022).

However, we note that in the worst case BanditMIPS may take $O(d)$ computations per atom when most atoms are equally good, for example in datasets where the atoms are symmetrically distributed around $\mathbf{q}$. For example, if each atom's coordinates are drawn i.i.d. from the *same* distribution, then the gaps Δ_i will scale inversely with d; to address this concern, we demonstrate how our algorithm maintains $O(1)$ scaling with respect to d in practice in Appendix 11.

Optimal weights for non-uniform sampling: Let $J \sim P_{\mathbf{w}}$ be a random variable following the categorical distribution $P_{\mathbf{w}}$, where $\mathbb{P}(J = j) = w_j \geq 0$ and $\sum_{j\in[d]} w_j = 1$. The arm parameter μ_i of an atom i can be estimated by the unbiased estimator $X_{iJ} = \frac{1}{dw_J} v_{iJ} q_J$. (Note that d is fixed and known in advance). To see that X_{iJ} is unbiased, we observe that $\mathbb{E}_{J\sim P_{\mathbf{w}}}[X_{iJ}] = \sum_{j\in[d]} w_j \frac{1}{dw_j} v_{ij} q_j = \sum_{j\in[d]} \frac{v_{ij}q_j}{d} = \mu_i$.

We are interested in finding the best weights $\mathbf{w}^*$, i.e., those that minimize the combined variance

$$\underset{w_1,\ldots,w_d\geq 0}{\arg\min} \sum_{i\in[n]} \mathrm{Var}_{J\sim P_{\mathbf{w}}}[X_{iJ}], \quad s.t. \sum_{j\in[d]} w_j = 1. \quad (3)$$

Theorem 2. *The solution to Problem* (3) *is*

$$w_j^* = \frac{\sqrt{q_j^2 \sum_{i\in[n]} v_{ij}^2}}{\sum_{j\in[d]} \sqrt{q_j^2 \sum_{i\in[n]} v_{ij}^2}}, \quad \textit{for } j = 1, \ldots, d. \quad (4)$$

The proof of Theorem 2 is provided in Appendix 8.

Remark: In practice, computing the atom variance $\sum_{i\in[n]} v_{ij}^2$ requires $O(nd)$ operations and can be computationally prohibitive. However, we may approximate $\sum_{i\in[n]} v_{ij}^2$ based on domain-specific assumptions. Specifically, if we assume that for each coordinate j, q_j has a similar magnitude as v_{ij}'s, we can approximate $\frac{1}{n}\sum_{i\in[n]} v_{ij}^2 \approx q_j^2$ and set $w_j^* = \frac{q_j^2}{\sum_{j\in[d]} q_j^2}$. In the non-uniform sampling versions of BanditMIPS, we use an additional hyperparameter β and let $w_j^* \propto q_j^{2\beta}$. β can be thought of as a temperature parameter which governs how uniformly (or not) we sample the coordinates based on the query vector's values. We note that $\beta = 1$ corresponds Equation (4).

The version we call BanditMIPS-α corresponds to taking the limit $\beta \to \infty$. In this case, we sort the query vector explicitly and sample coordinates in order of the sorted query vector; the sub-Gaussianity parameter used in BanditMIPS-α is then the same as that in the original problem with uniform sampling. While the sort incurs $O(d\log d)$ cost, we find this still improves the overall sample complexity of the algorithm relative to the closest baseline when $O(d\log d + n)$ is better than $O(n\sqrt{d})$, as is often the case in practice.

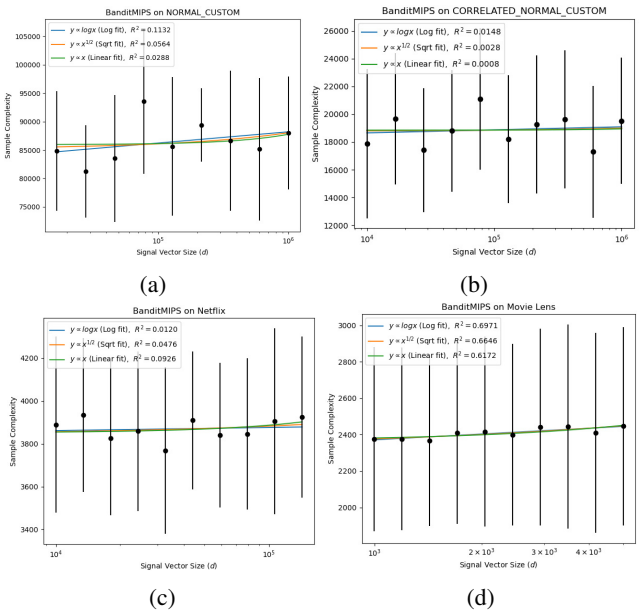


Figure 1: Sample complexity of BanditMIPS for different values of d on all four datasets. 95% CIs are provided around the mean and are computed from 10 random trials. The sample complexity of BanditMIPS does not scale with d. Note that the values of R^2, the coefficient of determination, are similar for linear, logarithmic, and square root fits, which suggests the scaling is actually constant.

5 Experiments

We empirically evaluate the performance of BanditMIPS and the non-uniform sampling version BanditMIPS-α on four synthetic and real-world datasets, comparing them to 8 state-of-art MIPS algorithms. We considered the two synthetic datasets, `NORMAL_CUSTOM` and `CORRELATED_NORMAL_CUSTOM`, to assess the performance across a wide parameter range. We further considered the two real-world datasets, the Netflix Prize dataset ($n = 6,000$, $d = 400,000$) (Bennett, Lanning, and Netflix 2007) and the Movie Lens dataset ($n = 4,000$, $d = 6,000$) (Harper and Konstan 2015), to provide additional evaluations. We compared our algorithms to 8 baseline MIPS algorithms: LSH-MIPS (Shrivastava and Li 2014a), H2-ALSH-MIPS (Huang et al. 2018), NEQ-MIPS (Dai et al. 2020), PCA-MIPS (Bachrach et al. 2014), BoundedME (Liu, Wu, and Mozafari 2019), Greedy-MIPS (Yu et al. 2017), HNSW-MIPS (Malkov and Yashunin 2016; Morozov and Babenko 2018b). and NAPG-MIPS (Tan et al. 2021). Throughout the experiments, we focus on the sample complexity, defined as the number of coordinate-wise multiplications performed. Appendix 9 provides additional details on our experimental settings.

Scaling with dimension d: We first assess the scaling with d for BanditMIPS on the four datasets. We subsampled features from the full datasets, evaluating d up to $1,000,000$ on simulated data and up to $400,000$ on real-world data. Results are reported in Figure 1. In all trials, BanditMIPS returns the

correct answer to MIPS. We determined that BanditMIPS did not scale with d in all experiments, validating our theoretical results on the sample complexity.

Comparison of sample complexity: We next compare the sample complexity of BanditMIPS and BanditMIPS-α to 8 state-of-art MIPS algorithms on the four datasets across different values of d. We only used a subset of up to 20K features because some of the baseline algorithms were prohibitively slow for larger values of d. Results are reported in Figure 2. We omit GREEDY-MIPS from Figure 2 because its sample complexity was significantly worse than all algorithms, and omit HNSW-MIPS as its performance was strictly worse than NAPG-MIPS (a related baseline). In measuring sample complexity, we measure *query-time* sample complexity, meaning we neglect the cost of preprocessing for the baseline algorithms which is favorable to the baselines. Nonetheless, our two algorithms substantially outperformed other algorithms on all four datasets, demonstrating their superiority in sample efficiency. For example, on the Movie Lens dataset, BanditMIPS and BanditMIPS-α are $20\times$ and $27\times$ faster than the closest baseline (NEQ-MIPS). In addition, the non-uniform sampling version BanditMIPS-α outperformed the default version BanditMIPS in 3 out 4 datasets, suggesting the weighted sampling technique further improves sample efficiency. BanditMIPS-α demonstrated slightly worse performance than BanditMIPS on the Netflix dataset, possibly because the highest-value coordinates for the randomly sampled query vectors had low dot products with the atoms.

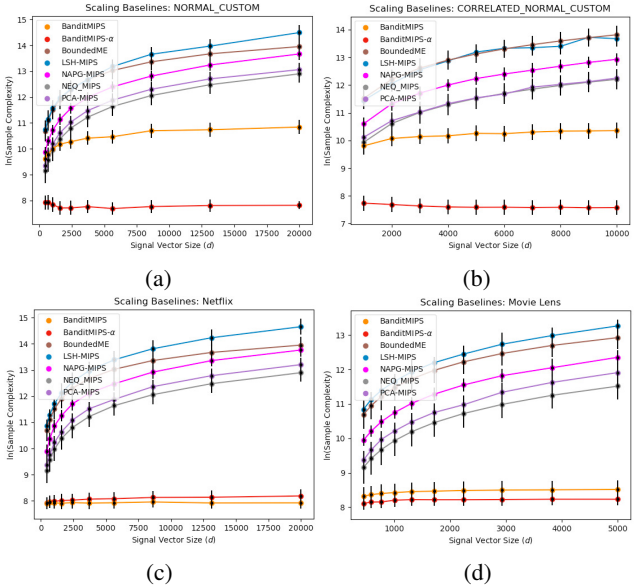


Figure 2: Comparison of sample complexity between BanditMIPS, BanditMIPS-α, and other baseline algorithms for different values of d across all four datasets. The y-axis is on a logarithmic scale. 95% CIs are provided around the mean and are computed from 10 random trials. BanditMIPS and BanditMIPS-α outperformed other baselines. For example, BanditMIPS achieves sample efficiency that surpasses the next best algorithm by up to $\times 30$ in the Movie Lens dataset.

Algorithms	Speedup
Naïve algorithm	1.00x
BoundedMe	0.41x
BanditMIPS	14.19x
BanditMIPS-α	9.93x

Table 2: Wall Clock Time Comparison on MNIST dataset. Experimental settings are as follows: ϵ=0.1, δ=0.1, number of atoms=1000, and signal vector size=6000.

Algorithms	Speedup
Naïve algorithm	1.00x
BoundedMe	0.36x
BanditMIPS	53.02x
BanditMIPS-α	25.97x

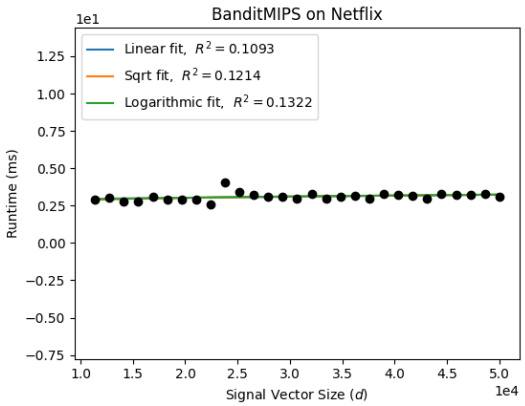
Table 3: Wall Clock Time Comparison on Movie Lens dataset. Experimental settings are as follows: ϵ=0.1, δ=0.1, number of atoms=1000, and signal vector size=100000.

Algorithms	Speedup
Naïve algorithm	1.00x
BoundedMe	1.02x
BanditMIPS	4.00x
BanditMIPS-α	6.02x

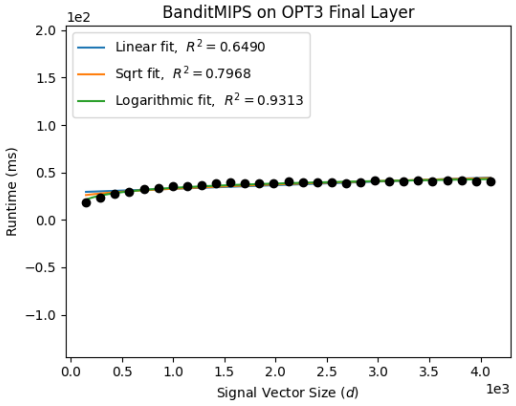
Table 4: Wall Clock Time Comparison on OPT-6.7B head dataset. OPT-6.7B head dataset refers to the MIPS task using the LM Head of the OPT-6.7B model as atoms and the final hidden vectors from randomly generated sentences as queries. Experimental settings are as follows: ϵ=1.0, δ=0.9, number of atoms=10000, and signal vector size=4096.

Wallclock speedup and scaling: We note that the sample complexity of BanditMIPS may not reliably predict its speedup for the MIPS problem, considering the highly optimized vector-vector dot product techniques. Thus, we offer wall-clock time comparisons as a complementary analysis, demonstrating that even with modest optimizations in Python, BanditMIPS significantly outperforms the next best algorithm (BoundedME). As shown in Table 2 and Table 3, BanditMIPS surpasses BoundedME by a large margin for the MNIST and Movie Lens dataset. We also employed BanditMIPS on the classification layer of OPT-6.7B where MIPS serves to efficiently determine the next token to generate. Table 4 demonstrates this impressive speedup achieved by BanditMIPS for dimension size 4096. All sampling-based algorithms successfully return an ϵ-suboptimal atom with $1-\delta$ probability for the given ϵ and δ. Additionally, figure 3 demonstrates that BanditMIPS exhibits $O(1)$ scaling with respect to dimensionality (d) in terms of wall-clock time on the Netflix and Movie Lens dataset.

Trade-off between speed and accuracy: We evaluate the trade-off between speed and accuracy by varying the error probability δ in our algorithm and the corresponding hyper-parameters in the baseline algorithms (see Appendix 9 for more details). As in (Liu, Wu, and Mozafari 2019), we define the speedup of an algorithm to be:

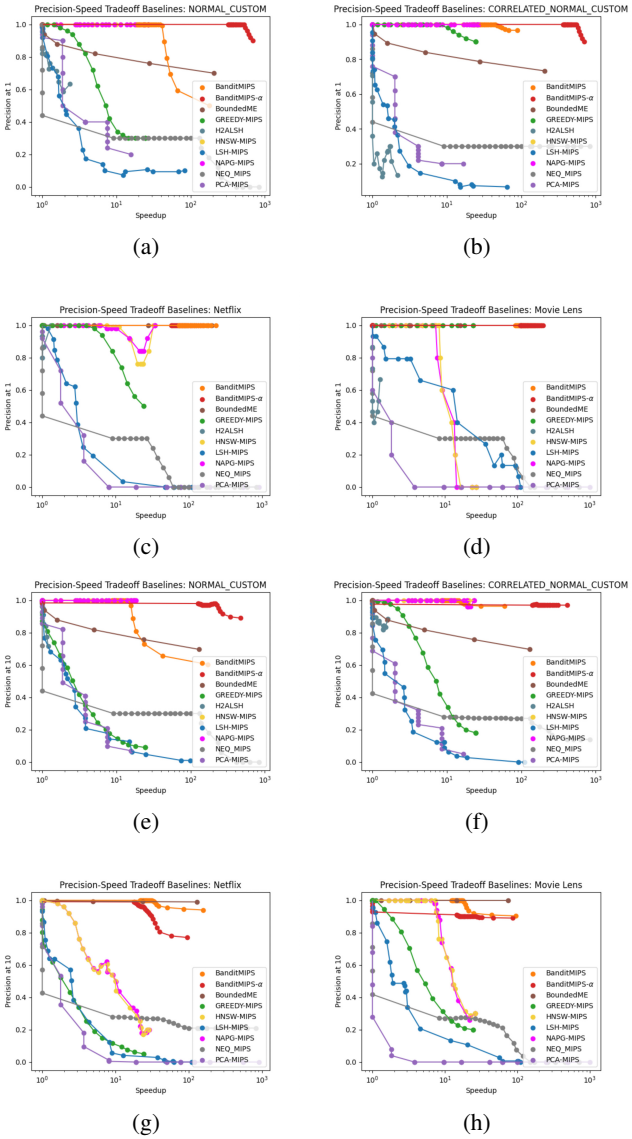


(a)



(b)

Figure 3: Wallclock time scaling of BanditMIPS for the Netflix dataset and OPT3-6.7b LM Head (with OPT3-6.7b model as atoms and final hidden vectors from randomly generated sentences as queries). The runtime of BanditMIPS is constant in d, as is expected. Means were calculated from 10 random seeds. For the Netflix dataset we have $\epsilon = 0.1$, $\delta = 0.1$, and 1000 atoms. For the LM Head, settings are $\epsilon = 1.0$, $\delta = 0.9$, and 1000 atoms.

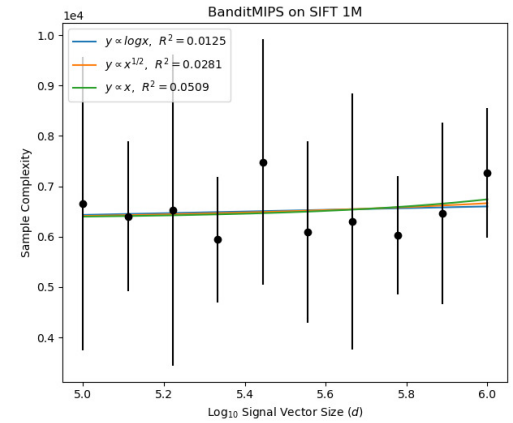


(a) (b) (c) (d) (e) (f) (g) (h)

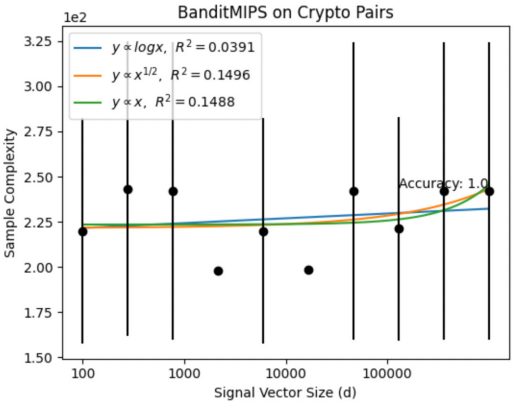
Figure 4: Trade-off between accuracy and speed for various algorithms across all four datasets. The x-axis represents the speedup relative to the naive $O(nd)$ algorithm and the y-axis shows the proportion of times an algorithm returned correct answer; higher is better. Each dot represents the mean across 10 random trials and the CIs are omitted for clarity. Our algorithms consistently achieve better accuracies at higher speedup values than the baselines. (a) through (d) is precision at $k = 1$ (i.e. the best arm) whereas (e) through (h) is precision at $k = 5$.

speedup $= \frac{\text{sample complexity of naïve algorithm}}{\text{sample complexity of compared algorithm}}$. The accuracy is defined as the proportion of times each algorithm returns the true MIPS solution. The tradeoff results for Normal Custom, Correlated Normal Custom, Netflix, and Movie Lens datasets are reported in Figure 4. Our algorithms achieved the best tradeoff on all four datasets, again demonstrating the superiority of our algorithms in efficiently and accurately solving the MIPS problem. Note that this figure also includes the k-MIPS setting where the goal is to find the top k atoms. In our particular case, $k = 5$. Even in this setting, our algorithms obtained a similar improvement over other baselines.

Real-world high-dimensional datasets: We also verify the $O(1)$ scaling with d on two real-world, high-dimensional datasets: the `Sift-1M` (Jégou, Douze, and Schmid 2011) and `CryptoPairs` datasets (Carsten 2022). The `Sift-1M` dataset consists of scale-invariant feature transform (Lowe 1999) features of 128 different images; each image is an atom with 1 million dimensions. The `CryptoPairs` dataset consists of the historical trading data of more than 400 trading pairs at 1 minute resolution reaching back until the year 2013. On these datasets, BanditMIPS appears to scale as $O(1)$ with d even to a million dimensions (Figure 6). This suggests

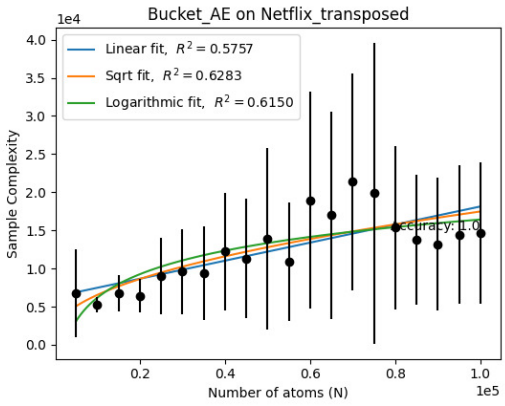


(a)

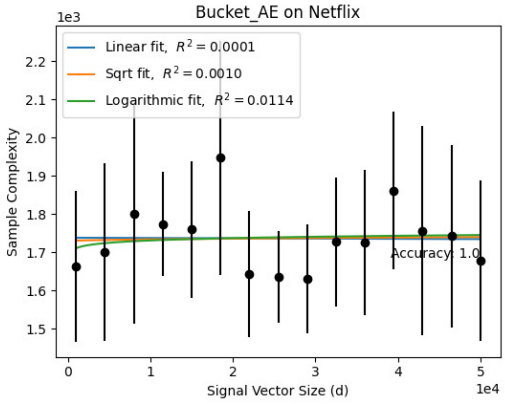


(b)

Figure 5: Sample complexity of BanditMIPS versus d for the `Sift-1M` and `CryptoPairs` datasets. BanditMIPS scales as $O(1)$ with respect to d for both datasets. Means and uncertainties were obtained by averaging over 5 random seeds. BanditMIPS returns the correct solution to MIPS in each trial.



(a)



(b)

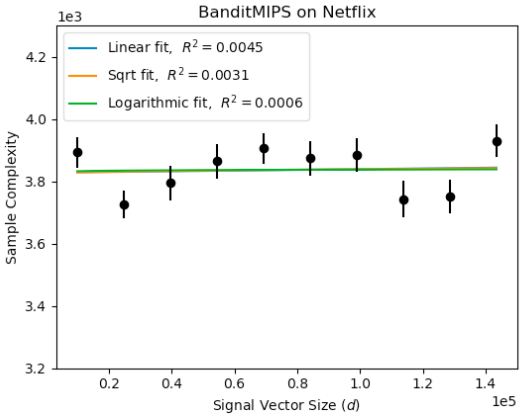
Figure 6: Sample complexity of `Bucket_AE` for both n and d on the `Netflix` dataset averaged over 5 random seeds. This demonstrates that BanditMIPS's constant scaling in dimension d is independent of optimizations deployed in the n-dimension, opening the door for many extensions of BanditMIPS with existing techniques.

that the necessary assumptions outlined in Sections 3 and 4 are satisfied on these real-world, high-dimensional datasets. Note that the high dimensionality of these datasets makes them prohibitively expensive to run scaling experiments as in Section 5 or the tradeoff experiments as in Section 5 for baseline algorithms.

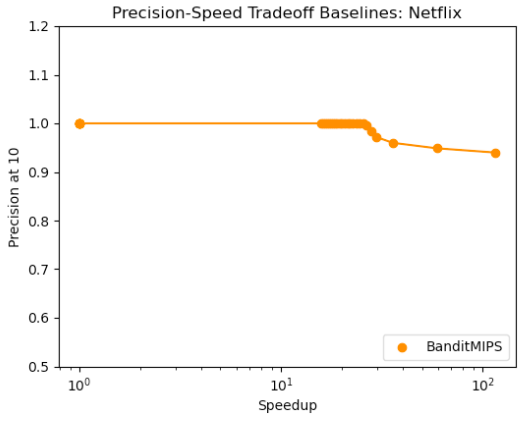
Preprocessing with BanditMIPS: BanditMIPS obviates any preprocessing in the dimension d as shown in our experiments above. An added benefit is that our algorithm also works *in conjunction* with preprocessing methods that reduce the scaling with respect to the number of atoms n. To show this compatibility, we implemented `Bucket_AE` which combines BanditMIPS with a normalized binning technique. More precisely, we estimate the norm of each atom with a constant number of samples to which we sort them into bins of b atoms in decreasing order (b is a hyperparameter). When running BanditMIPS, we make comparisons between only the best atoms in each bin and eliminate an entire bin if the maximum potential of that bin's best atom is less than the current largest sampled inner product across all bins. Intuitively, this allows us to filter atoms with small estimated norm more quickly. Indeed, figure 6b demonstrates that `Bucket_AE` reduces the scaling with n while maintaining $O(1)$ scaling with d on the real-world Netflix dataset. Furthermore, we observed that `Bucket_AE` returns the correct solution to BanditMIPS for all trials.

Robustness to ϵ-corruption: We end the experiments section with a discussion on BanditMIPS 's robustness to ϵ-corruption. Most works in the MIPS literature do not analyze the case that the data is corrupted by noise and provide no guarantees about robustness in this setting. However, BanditMIPS is actually robust to ϵ-corruption. More formally,

assume that each atom's coordinates are corrupted with noise drawn from a Gaussian with mean 0 and standard deviation ϵ. By the Central Limit Theorem, this noise will effectively vanish when averaged across many coordinates. For this reason, in our implementation, we set the batch size (which is the minimum number of coordinates sampled for any atom) to 100 or greater. As long as $\frac{\epsilon}{\text{batch size}\sigma} << 1$, i.e., the noise level ϵ is reasonable, BanditMIPS will still return the correct solution to the MIPS problem. We verify this on the Netflix dataset corrupted under this noise model, where ϵ is set to 20% of the maximum possible coordinate value of the dataset. We observe that BanditMIPS still scales as $O(1)$ with respect to d as shown in Figure 7 and returns correct solutions even at fairly high speedup values.



(a)



(b)

Figure 7: (a): Sample complexity of BanditMIPS versus dimension d for the corrupted Netflix dataset. Even in the uncorrupted settings, BanditMIPS maintains its O(1) scaling with respect to dimensionality (d). (b): Trade-off between top-10 accuracy and speed. The x-axis represents the speedup relative to the naive $O(nd)$ algorithm and the y-axis shows the proportion of times an algorithm returned correct answer; higher is better. Our algorithm returns an accurate solution even at high speedup values.

6 Conclusions and Limitations

Conclusions: In this work, we presented BanditMIPS and BanditMIPS-α, novel algorithms for the MIPS problem. In contrast with prior work, BanditMIPS requires no preprocessing of the data or that the data be nonnegative, and provides hyperparameters to trade off accuracy and runtime. BanditMIPS scales better to high-dimensional datasets under reasonable assumptions and outperformed the prior state-of-the-art significantly. BanditMIPS scales better to high-dimensional datasets under reasonable assumptions and outperformed the prior state-of-the-art significantly. Finally, we combined the approaches of BanditMIPS and BanditMIPS-α with other preprocessing techniques to reduce their scaling with n.
Limitations: Though the assumptions for BanditMIPS and BanditMIPS-α are often satisfied in practice, requiring them may be a limitation of our approach. In particular, when many of the arm gaps are small, BanditMIPS will compute the inner products for the relevant atoms naïvely.
Future directions: Future work may focus on relaxing these assumptions. Also, BanditMIPS has the potential to empower diverse machine learning algorithms with rich high-dimensional datasets, eliminating the necessity for dimensionality reduction. Subsequent exploration could delve into accelerating algorithms like Hierarchical Navigable Small World and a classification layer in a large language model.

References

Abuzaid, F.; Sethi, G.; Bailis, P.; and Zaharia, M. 2019. To Index or Not to Index: Optimizing Exact Maximum Inner Product Search. In 2019 IEEE 35th International Conference on Data Engineering (ICDE), 1250–1261. ISSN: 2375-026X.

Alexander, P.; Malkov, Y.; Andrey, L.; and Krylov, V. 2011. Approximate Nearest Neighbor Search Small World Approach.

Amagata, D.; and Hara, T. 2021. Reverse Maximum Inner Product Search: How to efficiently find users who would like to buy my item? Fifteenth ACM Conference on Recommender Systems, 273–281. Conference Name: RecSys '21: Fifteenth ACM Conference on Recommender Systems ISBN: 9781450384582 Place: Amsterdam Netherlands Publisher: ACM.

Amato, G.; and Savino, P. 2008. Approximate similarity search in metric spaces using inverted files. In Lempel, R.; Perego, R.; and Silvestri, F., eds., 3rd International ICST Conference on Scalable Information Systems, INFOSCALE 2008, Vico Equense, Italy, June 4-6, 2008, 28. ICST / ACM.

Andoni, A.; Indyk, P.; Laarhoven, T.; Razenshteyn, I.; and Schmidt, L. 2015. Practical and optimal LSH for angular distance. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., Advances in neural information processing systems, volume 28. Curran Associates, Inc.

Aouali, I.; Benhalloum, A.; Bompaire, M.; Ait Sidi Hammou, A.; Ivanov, S.; Heymann, B.; Rohde, D.; Sakhi, O.; Vasile, F.; and Vono, M. 2022. Reward Optimizing Recommendation using Deep Learning and Fast Maximum Inner Product Search. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, 4772–4773. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-9385-0.

Audibert, J.-y.; Bubeck, S.; and Munos, R. 2010. Best Arm Identification in Multiarmed Bandits. In In 23rd Annual Conference on Learning Theory.

Babenko, A.; and Lempitsky, V. 2012. The inverted multi-index. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, 3069–3076. ISSN: 1063-6919.

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, RecSys '14, 257–264. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-2668-1.

Bagaria, V.; Baharav, T. Z.; Kamath, G. M.; and David, N. T. 2021. Bandit-Based Monte Carlo Optimization for Nearest Neighbors. IEEE Journal on Selected Areas in Information Theory, 2(2): 599–610.

Bagaria, V.; Kamath, G. M.; Ntranos, V.; Zhang, M. J.; and Tse, D. 2018. Medoids in Almost-Linear Time via Multi-Armed Bandits. In International Conference on Artificial Intelligence and Statistics, 500–509.

Bagaria, V.; Kamath, G. M.; and Tse, D. 2018. Adaptive monte-carlo optimization. arXiv:1805.08321.

Baharav, T. Z.; Cheng, G.; Pilanci, M.; and Tse, D. 2022. Approximate Function Evaluation via Multi-Armed Bandits. In International Conference on Artificial Intelligence and Statistics, 108–135. PMLR.

Baharav, T. Z.; and Tse, D. 2019. Ultra Fast Medoid Identification via Correlated Sequential Halving. In Advances in Neural Information Processing Systems, 3650–3659.

Ballard, G.; Kolda, T. G.; Pinar, A.; and Seshadhri, C. 2015. Diamond Sampling for Approximate Maximum All-Pairs Dot-Product (MAD) Search. In 2015 IEEE International Conference on Data Mining, 11–20. Atlantic City, NJ, USA: IEEE. ISBN 978-1-4673-9504-5.

Bardenet, R.; and Maillard, O.-A. 2015. Concentration Inequalities for Sampling without Replacement. Bernoulli, 21(3): 1361–1385.

Bennett, J.; Lanning, S.; and Netflix, N. 2007. The Netflix Prize. In In KDD Cup and Workshop in Conjunction with KDD.

Bernhardsson, E. 2018. Annoy: Approximate Nearest Neighbors in C++/Python. Python package version 1.13.0.

Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; Driessche, G. B. V. D.; Lespiau, J.-B.; Damoc, B.; Clark, A.; Casas, D. D. L.; Guy, A.; Menick, J.; Ring, R.; Hennigan, T.; Huang, S.; Maggiore, L.; Jones, C.; Cassirer, A.; Brock, A.; Paganini, M.; Irving, G.; Vinyals, O.; Osindero, S.; Simonyan, K.; Rae, J.; Elsen, E.; and Sifre, L. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In Proceedings of the 39th International Conference on Machine Learning, 2206–2240. PMLR.

Boucheron, S.; Lugosi, G.; and Massart, P. 2013. Concentration Inequalities: A Nonasymptotic Theory of Independence.

Boytsov, L.; and Naidan, B. 2013a. Engineering Efficient and Effective Non-metric Space Library. In Brisaboa, N. R.; Pedreira, O.; and Zezula, P., eds., Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings, volume 8199 of Lecture Notes in Computer Science, 280–293. Springer.

Boytsov, L.; and Naidan, B. 2013b. Learning to Prune in Metric and Non-Metric Spaces. In Burges, C. J. C.; Bottou, L.; Ghahramani, Z.; and Weinberger, K. Q., eds., Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, 1574–1582.

Boytsov, L.; Novak, D.; Malkov, Y. A.; and Nyberg, E. 2016. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. In Mukhopadhyay, S.; Zhai, C.; Bertino, E.; Crestani, F.; Mostafa, J.; Tang, J.; Si, L.; Zhou, X.; Chang, Y.; Li, Y.; and Sondhi, P., eds., Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016, 1099–1108. ACM.

Brigham, E. O. 1988. The Fast Fourier Transform and Its Applications.

Bubeck, S.; Munos, R.; and Stoltz, G. 2011. Pure Exploration in Finitely-Armed and Continuous-Armed Bandits. Theoretical Computer Science, 412(19): 1832–1852.

Carsten. 2022. 400+ Crypto Currency Pairs at 1-Minute Resolution.

Catoni, O. 2012. Challenging the empirical mean and empirical variance: a deviation study. In Annales de l'IHP Probabilités et statistiques, volume 48, 1148–1185.

Chávez, E.; Figueroa, K.; and Navarro, G. 2008. Effective Proximity Retrieval by Ordering Permutations. IEEE Trans. Pattern Anal. Mach. Intell., 30(9): 1647–1658.

Chen, P. H.; Wei-cheng, C.; Hsiang-fu, Y.; Dhillon, I. S.; and Cho-jui, H. 2022. FINGER: Fast Inference for Graph-based Approximate Nearest Neighbor Search. ArXiv:2206.11408 [cs].

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.

Dai, X.; Yan, X.; Ng, K. K. W.; Liu, J.; and Cheng, J. 2020. Norm-Explicit Quantization: Improving Vector Quantization for Maximum Inner Product Search. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 51–58. ISSN: 2374-3468, 2159-5399 Issue: 01 Journal Abbreviation: AAAI.

Dasgupta, S. 2008. Random projection trees and low dimensional manifolds. 537–546.

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 2013 IEEE Conference on Computer Vision and Pattern Recognition, 1814–1821. Portland, OR, USA: IEEE. ISBN 978-0-7695-4989-7.

Ding, Q.; Yu, H.-F.; and Hsieh, C.-J. 2019. A Fast Sampling Algorithm for Maximum Inner Product Search. In Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, 3004–3012. PMLR. ISSN: 2640-3498.

Dong, W.; Wang, Z.; Charikar, M.; and Li, K. 2012. High-confidence near-duplicate image detection. In Ip, H. H.; and Rui, Y., eds., International Conference on Multimedia Retrieval, ICMR '12, Hong Kong, China, June 5-8, 2012, 1. ACM.

Donoho, D. L.; Tsaig, Y.; Drori, I.; and Starck, J.-L. 2012. Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit. IEEE Transactions on Information Theory, 58(2): 1094–1121.

Douze, M.; Jégou, H.; and Perronnin, F. 2016. Polysemous Codes. In Leibe, B.; Matas, J.; Sebe, N.; and Welling, M., eds., Computer Vision – ECCV 2016, Lecture Notes in Computer Science, 785–801. Cham: Springer International Publishing. ISBN 978-3-319-46475-6.

Even-Dar, E.; Mannor, S.; and Mansour, Y. 2002. PAC Bounds for Multi-armed Bandit and Markov Decision Processes. In Kivinen, J.; and Sloan, R. H., eds., Computational Learning Theory, Lecture Notes in Computer Science, 255–270. Berlin, Heidelberg: Springer. ISBN 978-3-540-45435-9.

Even-Dar, E.; Mannor, S.; and Mansour, Y. 2006. Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. The Journal of Machine Learning Research, 7: 1079–1105.

Even-Dar, E.; Mannor, S.; and Mansour, Y. 2006. Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. In Journal of Machine Learning Research, volume 7, 1079–1105.

Feng, C.; Lian, D.; Wang, X.; Liu, Z.; Xie, X.; and Chen, E. 2023. Reinforcement Routing on Proximity Graph for Efficient Recommendation. ACM Transactions on Information Systems, 41(1): 8:1–8:27.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2013. Optimized Product Quantization for Approximate Nearest Neighbor Search. 2946–2953.

Guo, R.; Geng, Q.; Simcha, D.; Chern, F.; Kumar, S.; and Wu, X. 2019. New Loss Functions for Fast Maximum Inner Product Search. ArXiv.

Guo, R.; Sun, P.; Lindgren, E.; Geng, Q.; Simcha, D.; Chern, F.; and Kumar, S. 2020. Accelerating large-scale inference with anisotropic vector quantization. In Proceedings of the 37th International Conference on Machine Learning, ICML'20, 3887–3896. JMLR.org.

Harper, F. M.; and Konstan, J. A. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interact. Intell. Syst., 5(4).

Hirata, K.; Amagata, D.; Fujita, S.; and Hara, T. 2022. Solving Diversity-Aware Maximum Inner Product Search Efficiently and Effectively. In Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, 198–207. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-9278-5.

Huang, Q.; Feng, J.; Zhang, Y.; Fang, Q.; and Ng, W. 2015. Query-Aware Locality-Sensitive Hashing for Approximate Nearest Neighbor Search. Proceedings of the VLDB Endowment, 9(1): 1–12.

Huang, Q.; Ma, G.; Feng, J.; Fang, Q.; and Tung, A. K. H. 2018. Accurate and Fast Asymmetric Locality-Sensitive Hashing Scheme for Maximum Inner Product Search. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, 1561–1570. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-5552-0.

Indyk, P.; and Motwani, R. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98, 604–613. New York, NY, USA: Association for Computing Machinery. ISBN 978-0-89791-962-3.

Jain, P.; and Kapoor, A. 2009. Active Learning for Large Multi-Class Problems. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, 762–769.

Jamieson, K.; Malloy, M.; Nowak, R.; and Bubeck, S. 2014. Lil' UCB : An Optimal Exploration Algorithm for Multi-Armed Bandits. In Proceedings of The 27th Conference on Learning Theory, 423–439. PMLR.

Jamieson, K.; and Nowak, R. 2014. Best-Arm Identification Algorithms for Multi-Armed Bandits in the Fixed Confidence Setting. In 2014 48th Annual Conference on Information Sciences and Systems (CISS), 1–6.

Jamieson, K.; and Talwalkar, A. 2016. Non-Stochastic Best Arm Identification and Hyperparameter Optimization. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 240–248. PMLR.

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.

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

Johnson, J.; Douze, M.; and Jégou, H. 2019. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3): 535–547.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product Quantization for Nearest Neighbor Search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(1): 117–128. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Jégou, H.; Tavenard, R.; Douze, M.; and Amsaleg, L. 2011. Searching in one billion vectors: Re-rank with source coding. Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on.

Kalyanakrishnan, S.; Tewari, A.; Auer, P.; and Stone, P. 2012. PAC Subset Selection in Stochastic Multi-Armed Bandits. In Proceedings of the 29th International Coference on International Conference on Machine Learning, ICML'12, 227–234. Madison, WI, USA: Omnipress. ISBN 978-1-4503-1285-1.

Karnin, Z.; Koren, T.; and Somekh, O. 2013. Almost Optimal Exploration in Multi-Armed Bandits. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, III–1238–III–1246. Atlanta, GA, USA: JMLR.org.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. Computer, 42(8): 30–37.

Lai, T. L.; and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. Advances in applied mathematics, 6(1): 4–22.

Li, W. H. Y. L. L. Y. S. Z., M.; and Wan, H. 2020. Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. Expert Systems with Applications.

Liu, J.; Yan, X.; Dai, X.; Li, Z.; Cheng, J.; and Yang, M.-C. 2020. Understanding and Improving Proximity Graph Based Maximum Inner Product Search. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 139–146. ISSN: 2374-3468, 2159-5399 Issue: 01 Journal Abbreviation: AAAI.

Liu, R.; Wu, T.; and Mozafari, B. 2019. A Bandit Approach to Maximum Inner Product Search. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19, 4376–4383. Honolulu, Hawaii, USA: AAAI Press. ISBN 978-1-57735-809-1.

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, 860–868. PMLR.

Lorenzen, S. S.; and Pham, N. 2021. Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search (Extended Abstract). In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, 4789–4793. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6.

Lowe, D. 1999. Object Recognition from Local Scale-Invariant Features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, 1150–1157 vol.2. IEEE. ISBN 978-0-7695-0164-2.

Lu, K.; and Kudo, M. 2021. AdaLSH: Adaptive LSH for Solving c-Approximate Maximum Inner Product Search Problem. IEICE Transactions on Information and Systems, E104.D(1): 138–145.

Lu, Z.; Hu, Y.; and Zeng, B. 2017. Sampling for Approximate Maximum Search in Factorized Tensor. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2400–2406. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3.

Ma, C.; Yu, F.; Yu, Y.; and Li, W. 2021. Learning Sparse Binary Code for Maximum Inner Product Search. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, 3308–3312. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8446-9.

Malkov, Y.; Ponomarenko, A.; Logvinov, A.; and Krylov, V. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. Inf. Syst., 45: 61–68.

Malkov, Y. A.; and Yashunin, D. A. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv e-prints, arXiv:1603.09320.

Matsui, Y.; Uchida, Y.; Jégou, H.; and Satoh, S. 2018. [Invited Paper] A Survey of Product Quantization. ITE Transactions on Media Technology and Applications, 6: 2–10.

Maurer, A.; and Pontil, M. 2009. Empirical Bernstein Bounds and Sample Variance Penalization. arXiv:0907.3740.

Morozov, S.; and Babenko, A. 2018a. Non-metric Similarity Graphs for Maximum Inner Product Search.

Morozov, S.; and Babenko, A. 2018b. Non-Metric Similarity Graphs for Maximum Inner Product Search. In Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc.

Mussmann, S.; and Ermon, S. 2016. Learning and Inference via Maximum Inner Product Search. In Proceedings of The 33rd International Conference on Machine Learning, 2587–2596. PMLR.

Naidan, B.; Boytsov, L.; and Nyberg, E. 2015. Permutation Search Methods are Efficient, Yet Faster Search is Possible. CoRR, abs/1506.03163.

Neyshabur, B.; and Srebro, N. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In Proceedings of the 32nd International Conference on Machine Learning, 1926–1934. PMLR.

Pham, N. 2020a. Sublinear Maximum Inner Product Search using Concomitants of Extreme Order Statistics. CoRR, abs/2012.11098.

Pham, N. 2021. Simple Yet Efficient Algorithms for Maximum Inner Product Search via Extreme Order Statistics. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, 1339–1347. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8332-5.

Pham, N. D. 2020b. Sublinear Maximum Inner Product Search using Concomitants of Extreme Order Statistics. ArXiv.

Ponomarenko, A.; Avrelin, N.; Naidan, B.; and Boytsov, L. 2014. Comparative Analysis of Data Structures for Approximate Nearest Neighbor Search.

Shrivastava, A.; and Li, P. 2014a. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In Advances in Neural Information Processing Systems, volume 27. Curran Associates, Inc.

Shrivastava, A.; and Li, P. 2014b. Improved Asymmetric Locality Sensitive Hashing (ALSH) for Maximum Inner Product Search (MIPS).

Shrivastava, A.; and Li, P. 2015. Improved Asymmetric Locality Sensitive Hashing (ALSH) for Maximum Inner Product Search (MIPS). In Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI'15, 812–821. Arlington, Virginia, USA: AUAI Press. ISBN 978-0-9966431-0-8.

Sivic, J.; and Zisserman, A. 2003. Video Google: A Text Retrieval Approach to Object Matching in Videos. volume 2, 1470–1477 vol.2.

Song, Y.; Gu, Y.; Zhang, R.; and Yu, G. 2021. ProMIPS: Efficient High-Dimensional c-Approximate Maximum Inner Product Search with a Lightweight Index. In 2021 IEEE 37th International Conference on Data Engineering (ICDE), 1619–1630. ISSN: 2375-026X.

Song, Z.; Xu, Z.; Yang, Y.; and Zhang, L. 2022. Accelerating Frank-Wolfe Algorithm using Low-Dimensional and Adaptive Data Structures. Publisher: arXiv Version Number: 1.

Sun, P.; Guo, R.; and Kumar, S. 2023. Automating Nearest Neighbor Search Configuration with Constrained Optimization. ArXiv:2301.01702 [cs].

Tan, S.; Xu, Z.; Zhao, W.; Fei, H.; Zhou, Z.; and Li, P. 2021. Norm Adjusted Proximity Graph for Fast Inner Product Retrieval. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, 1552–1560. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8332-5.

Tan, S.; Zhou, Z.; Xu, Z.; and Li, P. 2019. On Efficient Retrieval of Top Similarity Vectors. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 5236–5246. Hong Kong, China: Association for Computational Linguistics.

Tiwari, M.; Zhang, M. J.; Mayclin, J.; Thrun, S.; Piech, C.; and Shomorony, I. 2020. Banditpam: Almost linear time k-medoids clustering via multi-armed bandits. Advances in Neural Information Processing Systems, 33: 10211–10222.

Wu, G.; Zhu, B.; Li, J.; Wang, Y.; and Jia, Y. 2022. H2SA-ALSH: A Privacy-Preserved Indexing and Searching Schema for IoT Data Collection and Mining. Wireless Communications and Mobile Computing, 2022: e9990193. Publisher: Hindawi.

Wu, X.; Guo, R.; Kumar, S.; and Simcha, D. 2019. Local Orthogonal Decomposition for Maximum Inner Product Search. ArXiv.

Xiang, L.; Yan, X.; Lu, L.; and Tang, B. 2021. GAIPS: Accelerating Maximum Inner Product Search with GPU. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, 1920–1924. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8037-9.

Xu, Z.; Song, Z.; and Shrivastava, A. 2021. Breaking the Linear Iteration Cost Barrier for Some Well-known Conditional Gradient Methods Using MaxIP Data-structures. In Advances in Neural Information Processing Systems, volume 34, 5576–5589. Curran Associates, Inc.

Yan, X.; Li, J.; Dai, X.; Chen, H.; and Cheng, J. 2018. Norm-Ranging LSH for Maximum Inner Product Search. In Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc.

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, volume 30. Curran Associates, Inc.

Zhang, M.; Wang, W.; and He, Y. 2022. GraSP: Optimizing Graph-based Nearest Neighbor Search with Subgraph Sampling and Pruning. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM '22, 1395–1405. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-9132-0.

Zhang, M.; Zou, J.; and Tse, D. 2019a. Adaptive Monte Carlo Multiple Testing via Multi-Armed Bandits. In Proceedings of the 36th International Conference on Machine Learning, 7512–7522. PMLR.

Zhang, M.; Zou, J.; and Tse, D. 2019b. Adaptive Monte Carlo Multiple Testing via Multi-Armed Bandits. In International Conference on Machine Learning, 7512–7522.

Zhou, Z.; Tan, S.; Xu, Z.; and Li, P. 2019. Möbius Transformation for Fast Inner Product Search on Graph. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

7 Additional Related Work

In this appendix, we briefly describe other approaches attempt to reduce MIPS to a nearest neighbor search problem (NN). We note that the NN literature is extremely vast and has inspired the use of techniques based on permutation search (Naidan, Boytsov, and Nyberg 2015), inverted files (Amato and Savino 2008), vantage-point trees (Boytsov and Naidan 2013b), and more. The proliferation of NN algorithms has inspired several associated software packages (Bernhardsson 2018; Johnson, Douze, and Jégou 2019; Boytsov and Naidan 2013a) and tools for practical hyperparameter selection (Sun, Guo, and Kumar 2023). However, MIPS is fundamentally different from and harder than NN because the inner product is not a proper metric function (Morozov and Babenko 2018b). Nonetheless, NN techniques have inspired many direct approaches to MIPS, including those that rely on k-dimensional or random projection trees (Dasgupta 2008), concomitants of extreme order statistics (Pham 2020a, 2021, 2020b), ordering permutations (Chávez, Figueroa, and Navarro 2008), principle component analysis (PCA) (Bachrach et al. 2014), or hardware acceleration (Xiang et al. 2021; Abuzaid et al. 2019). All of these approaches require significant preprocessing that scales linearly in d, e.g., for computing the norms of the query or atom vectors, whereas BanditMIPS does not.

8 Proofs of Theorems

In this appendix, we present the proofs of Theorems 1 and 2.

Proof of Theorem 1:

Proof. Following the multi-armed bandit literature, we refer to each index i as an arm and refer to its optimization object μ_i as the arm parameter. We sometimes abuse the terminology and refer to the atom $\mathbf{v}_i$ as the arm, with the meaning clear from context. Pulling an arm corresponds to uniformly sampling a coordinate J and evaluating $v_{iJ}q_J$ and incurs an $O(1)$ computation. This allows us to focus on the number of arm pulls, which translates directly to coordinate-wise sample complexity.

First, we prove that with probability at least $1-\delta$, all confidence intervals computed throughout the algorithm are valid in that they contain the true parameter μ_i's. For a fixed atom $\mathbf{v}_i$ and a given iteration of the algorithm, the $\left(1-\frac{\delta}{2nd_{\text{used}}^2}\right)$ confidence interval satisfies

$$\Pr\left(|\mu_i-\hat{\mu}_i|>C_{d_{\text{used}}}\right)\leq 2e^{-C_{d_{\text{used}}}^2 d_{\text{used}}/2\sigma^2}\leq\frac{\delta}{2nd_{\text{used}}^2}$$

by Hoeffding's inequality and the choice of $C_{d_{\text{used}}}=\sigma\sqrt{\frac{2\log(4nd_{\text{used}}^2/\delta)}{d_{\text{used}}+1}}$. For a fixed arm i, for any value of d_{used} we have that the confidence interval is correct with probability at least $1-\frac{\delta}{n}$, where we used the fact that $1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots=\frac{\pi^2}{6}<2$. By another union bound over all n arm indices, all confidence intervals constructed by the algorithm are correct with probability at least $1-\delta$.

Next, we prove the correctness of BanditMIPS. Let $i^*=\arg\max_{i\in[n]}\mu_i$ be the desired output of the algorithm. First, observe that the main `while` loop in the algorithm can only run d times, so the algorithm must terminate. Furthermore, if all confidence intervals throughout the algorithm are valid, which is the case with probability at least $1-\delta$, i^* cannot be removed from the set of candidate arms. Hence, $\mathbf{v}_{i^*}$ (or some $\mathbf{v}_i$ with $\mu_i=\mu_{i^*}$) must be returned upon termination with probability at least $1-\delta$. This proves the correctness of Algorithm 1.

Finally, we examine the complexity of BanditMIPS. Let d_{used} be the total number of arm pulls computed for each of the arms remaining in the set of candidate arms at a given iteration in the algorithm. Note that for any suboptimal arm $i\neq i^*$ that has not left the set of candidate arms $\mathcal{S}_{\text{solution}}$, we must have $C_{d_{\text{used}}}\leq c_0\sqrt{\frac{\log(1/\delta)}{d_{\text{used}}}}$ by assumption (and this holds for our specific choice of $C_{d_{\text{used}}}$ in Algorithm 1). With $\Delta_i=\mu_{i^*}-\mu_i$, if $d_{\text{used}}>\frac{16c_0^2}{\Delta_i^2}\log\frac{n}{\delta\Delta_i}$, then

$$4C_{d_{\text{used}}}\leq 4c_0\sqrt{\frac{\log\frac{n}{\delta\Delta_i}}{d_{\text{used}}}}<\Delta_i$$

Furthermore,

$$\begin{aligned}\hat{\mu}_{i^*}-C_{d_{\text{used}}}&\geq\mu_{i^*}-2C_{d_{\text{used}}}\\&=\mu_i+\Delta_i-2C_{d_{\text{used}}}\\&>\mu_i+2C_{d_{\text{used}}}\\&>\hat{\mu}_i+C_{d_{\text{used}}}\end{aligned}$$

which means that i must be removed from the set of candidate arms by the end of that iteration.

Hence, the number of data point computations M_i required for any arm $i\neq i^*$ is at most

$$M_i\leq\min\left[\frac{16c_0^2}{\Delta_i^2}\log\frac{n}{\delta\Delta_i}+1,2d\right]$$

where we used the fact that the maximum number of computations for any arm is $2d$ when sampling with replacement. Note that bound this holds simultaneously for all arms i with probability at least $1-\delta$. We conclude that the total number of arm pulls M satisfies

$$M\leq\sum_{i\in[n]}\min\left[\frac{16c_0^2}{\Delta_i^2}\log\frac{n}{\delta\Delta_i}+1,2d\right]$$

with probability at least $1-\delta$.

As argued before, since each arm pull involves an $O(1)$ computation, M also corresponds to the total number of operations up to a constant factor. □

Proof of Theorem 2

Proof. Since all the X_{iJ}'s are unbiased, optimizing Problem (3) is equivalent to minimizing the combined second moment

$$\sum_{i\in[n]} \mathbb{E}_{J\sim P_{\mathbf{w}}}[X_{iJ}^2] = \sum_{i\in[n]}\sum_{j\in[d]} \frac{1}{d^2 w_j} q_j^2 v_{ij}^2 \tag{5}$$

$$= \sum_{j\in[d]} \left(\frac{1}{d^2 w_j} q_j^2 \sum_{i\in[n]} v_{ij}^2 \right). \tag{6}$$

The Lagrangian is given by

$$\mathcal{L}(\mathbf{w},\nu) = \sum_{j\in[d]} \left(\frac{1}{d^2 w_j} q_j^2 \sum_{i\in[n]} v_{ij}^2 \right) + \nu \left(1 - \sum_{j\in[d]} w_j \right). \tag{7}$$

Furthermore, the derivatives are

$$\frac{\partial \mathcal{L}(\mathbf{w},\nu)}{\partial w_j} = -\frac{q_j^2 \sum_{i\in[n]} v_{ij}^2}{d^2 w_j^2} - \nu \tag{8}$$

$$\frac{\partial \mathcal{L}(\mathbf{w},\nu)}{\partial \mu} = 1 - \sum_{j\in[d]} w_j. \tag{9}$$

By the Karush-Kuhn-Tucker (KKT) conditions, setting the derivatives to 0 gives

$$w_j^* = \frac{\sqrt{q_j^2 \sum_{i\in[n]} v_{ij}^2}}{\sum_{j\in[d]} \sqrt{q_j^2 \sum_{i\in[n]} v_{ij}^2}} \quad \text{for } j = 1,\ldots,d. \tag{10}$$

□

9 Description of Datasets

Here, we provide a more detailed description of the datasets used in our experiments.

Synthetic Datasets

In the `NORMAL_CUSTOM` dataset, a parameter θ_i is drawn for each atom from a standard normal distribution, then each coordinate for that atom is drawn from $\mathcal{N}(\theta_i, 1)$. The signals are generated similarly.

In the `CORRELATED_NORMAL_CUSTOM` dataset, a parameter θ is for the signal $\mathbf{q}$ from a standard normal distribution, then each coordinate for that signal is drawn from $\mathcal{N}(\theta, 1)$. Atom $\mathbf{v}_i$ is generated by first sampling a random weight $w_i \sim \mathcal{N}(0,1)$; then atom $\mathbf{v}_i$ is set to $w_i\mathbf{q}$ plus Gaussian noise.

Note that for the synthetic datasets, we can vary n and d. The values of n and d chosen for each experiment are described in Subsection 9.

Real-world datasets

Netflix Dataset: We use a subset of the data from the Netflix Prize dataset (Bennett, Lanning, and Netflix 2007) that contains the ratings of 6,000 movies by 400,000 customers. We impute missing ratings by approximating the data matrix via a low-rank approximation. Specifically, we approximate the data matrix via a 100-factor SVD decomposition. The movie vectors are used as the query vectors and atoms and d corresponds to the number of subsampled users.

Movie Lens Dataset: We use Movie Lens-1M dataset (Harper and Konstan 2015), which consists of 1 million ratings of 4,000 movies by 6,000 users. As for the Netflix dataset, we impute missing ratings by obtaining a low-rank approximation to the data matrix. Specifically, we perform apply a Non-negative Matrix Factorization (NMF) with 15 factors to the dataset to impute missing values. The movie vectors are used as the query vectors and atoms, with d corresponding to the number of subsampled users.

We note that for all datasets, the coordinate-wise inner products are sub-Gaussian random variables. In particular, this means the assumptions of Theorem 1 are satisfied and we can construct confidence intervals that scale as $O\left(\sqrt{\frac{\log 1/\delta'}{d'}}\right)$. We describe the setting for the sub-Gaussianity parameters in Section 9.

Experimental Settings

Scaling Experiments: In all scaling experiments, δ and ϵ were both set to 0.001 for BanditMIPS and BanditMIPS-α. ϵ is the hyperparameter in bandit algorithms that controls how far the returned arm is from the true optimal arm, allowing for an ϵ-suboptimal choice. For the `NORMAL_CUSTOM` and `CORRELATED_NORMAL_CUSTOM` datasets, the sub-Gaussianity parameter was set to 1. For the Netflix and Movie Lens datasets, the sub-Gaussianity parameter was set to 25. For the `CryptoPairs`, `SIFT-1M`, and `SimpleSong` datasets described in Appendix 10, the sub-Gaussianity parameters were set to $2.5e9$, $6.25e5$, and 25, respectively. The number of atoms was set to 100 and all other atoms used default values of hyperparameters for their sub-Gaussianity parameters.

Tradeoff Experiments: For the tradeoff experiments, the number of dimensions was fixed to $d = 10,000$. The various values of speedups were obtained by varying the hyperparameters of each algorithm. For NAPG-MIPS and HNSW-MIPS, for example, M was varied from 4 to 32, $ef\_constructions$ was varied from 2 to 500, and $ef\_searches$ was varied from 2 to 500. For Greedy-MIPS, $budget$ varied from 2 to 999. For LSH-MIPS, the number of hash functions and hash values vary from 1 to 10. For H2ALSH, δ varies from $\frac{1}{2^4}$ to $\frac{1}{2}$, c_0 varies from 1.2 to 5, and c varies from 0.9 to 2. For NEQ-MIPS, the number of codewords and codebooks vary from 1 to 100. For BanditMIPS , BanditMIPS-α, and BoundedME, speedups were obtained by varying δ from $\frac{1}{10^{10}}$ to 0.99 and ϵ from $\frac{1}{10^{10}}$ to 3. In our code submission, we include a one-line script to reproduce all of our results and plots.

All experiments were run on a 2019 Macbook Pro with a 2.4 GHz 8-Core Intel Core i9 CPU, 64 GB 2667 MHz DDR4 RAM, and an Intel UHD Graphics 630 1536 MB graphics card. Our results, however, should not be sensitive to hardware, as we used hardware-independent performance

Table 5: Frequencies for various musical notes.

Note	Frequency (Hz)
C4	256
E4	330
G4	392
C5	512
E5	660
G5	784

metrics (the number of coordinate-wise multiplications) for our results.

10 Application to Matching Pursuit on the `SimpleSong` Dataset

We construct a simple synthetic dataset, titled the `SimpleSong` Dataset where the query and atoms are audio signals sampled at 44,100 Hz and each coordinate value represents the signal's amplitude at a given point in time. Common musical notes are represented as periodic sine waves with the frequencies given in Table 5.

The query in this dataset is a simple song. The song is structured in 1 minute intervals, where the first interval – called an A interval – consists of a C4-E4-G4 chord and the second interval – called a B interval – consists of a G4-C5-E5 chord. The song is then repeated t times, bringing its total length to $2t$ minutes. The dimensionality of the the signal is $d = 2t * 44,100 = 88,200t$. The weights of the C4, E4, and G4 waves in the A intervals and the G4, C5, and E5 waves in the B intervals are in the ratio 1:2:3:3:2.5:1.5.

The atoms in this dataset are the sine waves corresponding to the notes with the frequencies show in Table 5, as well as notes of other frequencies.

The Matching Pursuit problem (MP) is a problem in which a vector $\mathbf{q}$ is approximated as a linear combination of the atoms $\mathbf{v}_1, \ldots, \mathbf{v}_n$. A common algorithm for MP involves solving MIPS to find the atom $\mathbf{v}_{i^*}$ with the highest inner product with the query, subtracting the component of the query parallel to $\mathbf{v}_{i^*}$, and re-iterating this process with the residual. Such an approach solves MIPS several times as a subroutine. Thus, an algorithm which accelerates MIPS should also then accelerate MP.

In the audio domain, we note that when the atoms $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are periodic functions with predefined frequencies, MP becomes a form of Fourier analysis in which the atoms are the Fourier components and their inner products with the query correspond to Fourier coefficients. For more detailed background on Fourier theory, we refer the reader to (Brigham 1988).

For convenience, we restrict t to be an integer in our experiments so a whole number of AB intervals are completed. We ran BanditMIPS with $\delta = \frac{1}{10,000}$ and $\sigma^2 = 6.25$ over 3 random seeds for various values of t. BanditMIPS is correctly able to recover the notes played in the song in order of decreasing strength: G4, C5, E4, E5, and C4 in each experiment.

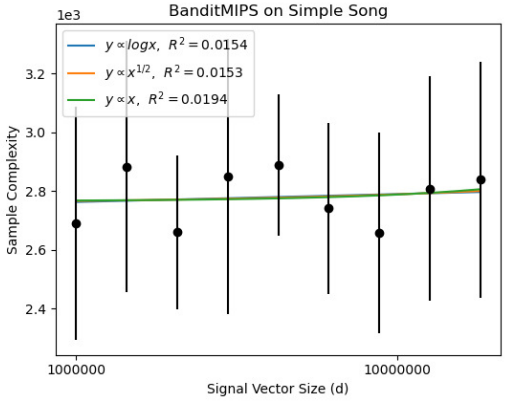


Figure 8: Sample complexity of MP when using BanditMIPS as a subroutine for MIPS on the `SimpleSong` dataset. The complexity of solving the problem does not scale with the length of the song, d. Uncertainties and means were obtained from 3 random seeds. BanditMIPS returns the correct solution to MIPS in each trial.

Furthermore, BanditMIPS is able to calculate their Fourier coefficients correctly. Crucially, the complexity of BanditMIPS to identify these components does not scale with d, the length of the song. Figure 8 demonstrates the total sample complexity of BanditMIPS to identify the first five Fourier components (five iterations of MIPS) of the song as the song length increases.

Our approach may suggest an application to Fourier transforms, which aim to represent signals in terms of constituent signals with predetermined set of frequencies. We acknowledge, however, that Fourier analysis is a well-developed field and that further research is necessary to compare such a method to state-of-the-art Fourier transform methods, which may already be heavily optimized or sampling-based.

11 BanditMIPS on a Highly Symmetric Dataset

In this section, we discuss a dataset on which the assumptions in Section 4 fail, namely when Δ scales with d. In this setting, BanditMIPS does not scale as $O(1)$ and instead scales linearly with d, as is expected.

We call this dataset the `SymmetricNormal` dataset. In this dataset, the signal has each coordinate drawn from $\mathcal{N}(0, 1)$ and each atom's coordinate is drawn i.i.d. from $\mathcal{N}(0, 1)$. Note that all atoms are therefore symmetric *a priori*.

We now consider the quantity $\Delta_{i,j}(d) \coloneqq \mu_1(d) - \mu_2(d)$, i.e., the gap between the first and second arm, where our notation emphasizes we are studying each quantity as d increases. Note that $\Delta_{i,j}(d) = \frac{\mathbf{v}_1^T q - \mathbf{v}_2^T q}{d}$. By the Central Limit Theorem, the sequence of random variables $\sqrt{d}\Delta_{i,j}(d)$ converges in distribution to $\mathcal{N}(0, \sigma_{i,j}^2)$ for some constant $\sigma_{i,j}^2$. Crucially, this implies that $\Delta_{i,j}(d)$ is on the order of $\frac{1}{\sqrt{d}}$.

The complexity results from Theorem 1 then predicts that BanditMIPS scales linearly with d. Indeed, this is what we

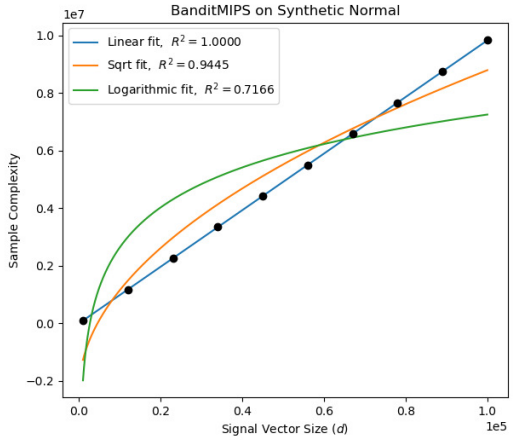


Figure 9: Sample complexity of BanditMIPS on the `SymmetricNormal` dataset. The sample complexity of BanditMIPS is linear with d, as is expected. Uncertainties and means were obtained from 10 random seeds.

observe in Figure 9.

In practice, this case can be dealt with by allowing for an ϵ-suboptimal atom vector to be returned. In this case, BanditMIPS will no longer depend on the Δ_i's for large d, and instead on the relative error hyperparameter ϵ. This is depicted in figure 10.

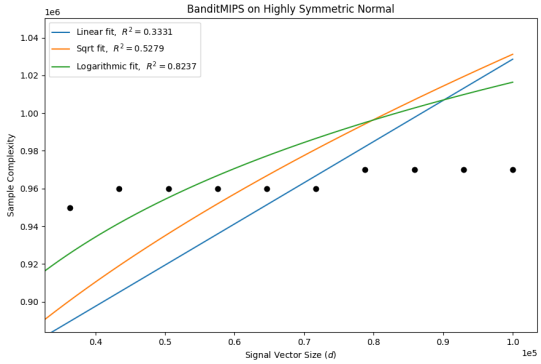


Figure 10: Sample Complexity of BanditMIPS on the Highly Symmetric Normal Dataset as a Function of d, allowing for ϵ-suboptimal atoms to be identified, with $\epsilon = 0.1$. The sample complexity of BanditMIPS scales as $O(1)$ with respect to d, even when all atoms have an equal inner product with the query vector.