Faster Maximum Inner Product Search in High Dimensions

Mo Tiwari [1] **Ryan Kang** [*1] **Je-Yong Lee** [*2] **Donghyun Lee** [*3] **Chris Piech** [1] **Sebastian Thrun** [1]
Ilan Shomorony [†4] **Martin Jinye Zhang** [†5]

Abstract

Maximum Inner Product Search (MIPS) is a ubiquitous task in machine learning applications. Given a query vector and n other vectors in d dimensions, the MIPS problem is to find the atom that has the highest inner product with the query vector. Existing MIPS algorithms scale at least as $O(\sqrt{d})$ with respect to d, which becomes computationally prohibitive in high-dimensional settings. In this work, we present BanditMIPS, a novel randomized algorithm that provably improves the state-of-the-art complexity from $O(\sqrt{d})$ to $O(1)$ with respect to d. We validate the scaling of BanditMIPS and demonstrate that BanditMIPS outperforms prior state-of-the-art MIPS algorithms in sample complexity, wall-clock time, and precision/speedup tradeoff across a variety of experimental settings. Furthermore, we propose a variant of our algorithm, named BanditMIPS-α, which improves upon BanditMIPS by employing non-uniform sampling across coordinates. We also demonstrate the usefulness of BanditMIPS in problems for which MIPS is a subroutine, including Matching Pursuit and Fourier analysis. Finally, we demonstrate that BanditMIPS can be used in conjunction with preprocessing techniques to improve its complexity with respect to n. All of our experimental results are reproducible via a 1-line script at `github.com/ThrunGroup/BanditMIPS`.

[*]Equal contribution [1]Department of Computer Science, Stanford University, Stanford, CA [2]Oxford University [3]University College London [4]Electrical and Computer Engineering, University of Illinois at Urbana-Champaign [5]Ray and Stephanie Lane Computational Biology Department, Carnegie Mellon University. Correspondence to: Mo Tiwari <motiwari@stanford.edu>, Ilan Shomorony <ilans@illinois.edu>, Martin Jinye Zhang <martinzh@andrew.cmu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

The Maximum Inner Product Search problem (MIPS) is a ubiquitous task that arises in many contexts, such as information retrieval, augmenting large auto-regressive language models, and the Matching Pursuit (MP) problem (Shrivastava & Li, 2014; Neyshabur & Srebro, 2015; Yu et al., 2017; Locatello et al., 2017; Sivic & Zisserman, 2003; Dong et al., 2012; Boytsov et al., 2016; Borgeaud et al., 2022).

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 & 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 scales as $O(nd)$, as it evaluates all nd scalar products. Significant prior work has focused on the improving the scaling with n, often via heavy preprocessing (Morozov & Babenko, 2018b; Liu et al., 2020); these approaches are discussed in detail in Section 1.1. However, fairly little work has focused on improving the complexity of MIPS algorithms with respect to d; most existing MIPS algorithms computationally prohibitive in high-dimensional datasets in domains like e-commerce, genomics, and finance.

In this work, we propose BanditMIPS, a state-of-the-art randomized algorithm for the MIPS problem that scales as $O(1)$ with respect to d, under general assumptions that often hold in practice. Intuitively, BanditMIPS performs adaptive, coordinate-wise sampling to estimate the inner products $\mathbf{v}_i^T \mathbf{q}$ for each i. BanditMIPS allocates more samples to "promising" atoms while simultaneously discarding "unpromsing" atoms quickly. The specific adaptive sampling procedure is motivated by multi-armed bandits (MAB) (Even-Dar et al., 2006).

Our specific contributions are as follows:

- In Section 3, we propose BanditMIPS, a novel state-of-the-art algorithm for MIPS in high-dimensional settings. We also propose BanditMIPS-α, which provides additional runtime speedup by sampling coordinates intelligently.
- In Section 4, we prove that BanditMIPS achieves $O(1)$ sample complexity with respect to d under general settings.
- In Section 5, we empirically demonstrate that BanditMIPS outperforms baseline algorithms across a variety of experimental settings and achieves up to a $30\times$ wall-clock time improvement over the next fastest algorithm. We also discuss how BanditMIPS can be used as a subroutine in other applications, such as Matching Pursuit and in the classification layer of a large language model (LLM).

We conclude in Section 6 with a discussion of limitations and future work.

1.1. Related work

The MIPS problem is well-studied and has inspired many algorithms due to its ubiquity. Prior work often assumes that the vector entries are nonnegative, performs non-adaptive sampling (Lu et al., 2017; Lorenzen & Pham, 2021; Ding et al., 2019), or relies on product quantization (Dai et al., 2020). A large family of MIPS algorithms are based on locality-sensitive hashing (LSH) (Song et al., 2021; Lu & Kudo, 2021; Wu et al., 2022; Ma et al., 2021) and proximity graphs, such as ip-NSW (Morozov & Babenko, 2018b). 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).

Other approaches attempt to reduce MIPS to a nearest neighbor search (NN) problem. The NN literature is vast and has inspired the use of techniques based on permutation search (Naidan et al., 2015), inverted files (Amato & Savino, 2008), vantage-point trees (Boytsov & Naidan, 2013b), k-dimensional or random projection trees (Dasgupta & Freund, 2008), concomitants of extreme order statistics (Pham, 2020a; 2021; 2020b), ordering permutations (Chávez et al., 2008), principle component analysis (PCA) (Bachrach et al., 2014), or hardware acceleration (Xiang et al., 2021; Abuzaid et al., 2019). The proliferation of NN algorithms has also inspired several associated software packages (Bernhardsson, 2018; Johnson et al., 2019; Boytsov & Naidan, 2013a) and tools for practical hyperparameter selection (Sun et al., 2023). 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. Furthermore, MIPS is fundamentally different from and harder than NN because the inner product is not a proper metric function (Morozov & Babenko, 2018a).

Recent work to improve scaling with d attempts to use dimensionality reduction techniques, but these methods often discard valuable information, particularly in high-dimensional settings where signal is diffuse, and usually scale linearly with d (Li et al., 2020). As such, some recent work attempts to improve scaling with d.

Perhaps most similar to our work is the BoundedMe, which solves the MIPS problem using an adaptive sampling approach (Liu et al., 2019). However, this method still scales as $O(\sqrt{d})$ with respect to d. Intuitively, BoundedMe is only adaptive to the relative *ranking* of the inner products; the number of times each atom is sampled does not adapt to the actual *values* of the sampled inner products. This approach is wasteful because information contained in the sampled inner products' values is discarded.

Multi-armed bandits: BanditMIPS is motivated by the best-arm identification problem in multi-armed bandits (MABs) (Audibert et al., 2010; Jamieson & Nowak, 2014; Jamieson et al., 2014; Jamieson & Talwalkar, 2016). In the best-arm identification 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 objective is to identify the arm with the largest reward 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., 2018a;b; Zhang et al., 2019b; Bagaria et al., 2021).

2. Preliminaries and Notation

We consider a query vector $\mathbf{q} \in \mathbb{R}^d$ and n atoms $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$. Let $[n]$ denote the set $\{1, \ldots, n\}$. 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* of atom $\mathbf{v}_i$ with $\mathbf{q}$. (Intuitively, if each of the coordinates of the atom $\mathbf{v}_i$ and $\mathbf{q}$ are drawn i.i.d., then the unnormalized inner product will scale with d, whereas the normalized inner product will not). 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 $\Delta := \min_{i \neq i^*} \Delta_i$. In this work, we primarily focus on the computational complexity of MIPS with respect to d.

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 $1-\delta$

3. Algorithm

We now discuss the reduction of the MIPS problem to a best-arm identification problem in the multi-armed bandits framework. Table 1 represents how BanditMIPS can be represented as a best-arm identification problem. Intuitively, we view each atom $\mathbf{v}_i$ as an arm with arm parameter $\mu_i := \frac{\mathbf{v}_i^T\mathbf{q}}{d}$. Pulling arm i corresponds to randomly sampling a coordinate $J \sim \text{Unif}[d]$ and evaluating the (scalar) product $X_i = q_J v_{iJ}$. Using this reduction, the best atom can be estimated using best-arm identification algorithms.

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-Gaussianity parameter σ
Output: $i^* = \arg\max_{i \in [n]} \mathbf{q}_i^T \mathbf{v}$

$\mathcal{S}_{\text{solution}} \leftarrow [n]$
$d_{\text{used}} \leftarrow 0$
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: $C_{d_{\text{used}}} \leftarrow \sigma\sqrt{\frac{2\log\left(4nd_{\text{used}}^2/\delta\right)}{d_{\text{used}}+1}}$
  end for
  $\mathcal{S}_{\text{solution}} \leftarrow \{i : \hat{\mu}_i + C_{d_{\text{used}}} \geq \max_{i'} \hat{\mu}_{i'} - C_{d_{\text{used}}}\}$
  $d_{\text{used}} \leftarrow d_{\text{used}} + 1$
end while
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 $i^* = \arg\max_{i \in \mathcal{S}_{\text{solution}}} \hat{\mu}_i$

With this reduction in mind, we propose BanditMIPS in Algorithm 1. BanditMIPS can be viewed as a combination of the Upper Confidence Bound (UCB) and successive elimination algorithms (Lai & Robbins, 1985; Even-Dar et al., 2006; Zhang et al., 2019a), applied to the MIPS problem. 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-Gaussianity parameter and possible relaxations of this assumption in Sections 3.2 and 4.

3.1. 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 $\mathbf{q}$. Motivated by this observation, we propose BanditMIPS-α, in which 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.

Warm start increases speed: In some settings, it may be necessary to solve the MIPS problem for a batch of m queries instead of just a single query. In such cases, we may 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 simultaneously. Such a procedure will eliminate the less promising atoms and avoid repeated sampling for each of the m MIPS problems, thereby improving 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.

3.2. 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 σ-sub-Gaussian with $\sigma^2 = \frac{b^2-a^2}{4}$. This is commonly the case, e.g., in recommendation systems where user ratings (each element of the query and atoms) are integers between 0 and 5. In such settings, we use this implied value of σ in our experiments in Section 5.

The σ-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. The value $\sigma^2 = \frac{b^2-a^2}{4}$ 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 (Maurer & Pontil, 2009).

4. Theoretical Analysis

We now present theoretical results on the correctness and computational complexity of BanditMIPS. Since each coordinate-wise multiplication only incurs $O(1)$ computational overhead to update running means and confidence intervals in Algorithm 1, sample complexity bounds translate directly to wall-clock times bounds via constant factors. For this reason, we focus on sample complexity bounds, in line with prior work (Tiwari et al., 2020; Bagaria et al., 2018b).

We present our main result in Theorem 4.1. In Theorem 4.1, 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 described in Section 3.2 satisfy this property.

Theorem 4.1. *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 4.1 is proven in the Appendix A. We note that c_0 is the sub-Gaussianity parameter described in Section 3.2 and is a constant. Intuitively, Theorem 4.1 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 $\mathbf{v}_i$, $\min\left[\frac{16c_0^2}{\Delta_i^2}\log\left(\frac{n}{\delta\Delta_i}\right) + 1, 2d\right]$, 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 4.1 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 3.2, this is a rather general assumption, 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. Our experiments in Section 5 also verify that this assumption holds in many real-world datasets.

Discussion of the hyperparameter δ: The hyperparameter δ allows users to trade off accuracy and sample complexity 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 4.1 provides an analysis of the effect of δ and we discuss appropriate ways to tune it in Section 5. We note that setting $\delta = 0$ reduces Algorithm 1 to the naïve algorithm for MIPS. In particular, Algorithm 1 is never worse in 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 prior work (e.g., Appendix 2 of Bagaria et al. (2018a)), this is the case under a wide range of distributional assumptions on the μ_is, e.g., when the μ_is 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, provided 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_is 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., 2018a; Baharav & Tse, 2019; Tiwari et al., 2020; Bagaria et al., 2021; Baharav et al., 2022).

We note that, in the worst case, BanditMIPS may take $O(d)$ computations per atom when most atoms are equally good, e.g., 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; We can ameliorate this problem by permitting an ϵ-suboptimal arm to be identified; we demonstrate how our algorithm still exhibits $O(1)$ scaling with respect to d with this modification in Appendix C.

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 4.2. *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 4.2 is provided in Appendix A.

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 knowledge. Specifically, if we assume that for each coordinate j, q_j has a similar magnitude to the 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.

5. Experiments

We now demonstrate the improvements of BanditMIPS and BanditMIPS-α over prior state-of-the-art across a variety of datasets and applications.

Datasets: We empirically evaluate the performance of BanditMIPS and BanditMIPS-α on two synthetic and four real-world datasets. The two synthetic datasets are called the `NORMAL_CUSTOM` and `CORRELATED_NORMAL_CUSTOM` datasets and are described in greater detail in Appendix B. We also conduct experiments on four real-world datasets, the Netflix Prize dataset (Bennett et al., 2007) the Movie Lens dataset (Harper & Konstan, 2015), the `Sift-1M` (Jégou et al., 2011) and `CryptoPairs` datasets (Carsten, 2022) to provide practical evaluations. We also discuss these datasets in greater detail in Appendix B.

Baseline MIPS algorithms: We compare BanditMIPS and BanditMIPS-α to eight baseline MIPS algorithms: LSH-MIPS (Shrivastava & Li, 2014), H2-ALSH-MIPS (Huang et al., 2018), NEQ-MIPS (Dai et al., 2020), PCA-MIPS (Bachrach et al., 2014), BoundedME (Liu et al., 2019), Greedy-MIPS (Yu et al., 2017), HNSW-MIPS (Malkov & Yashunin, 2016; Morozov & Babenko, 2018a). and NAPG-MIPS (Tan et al., 2021).

Metrics: Throughout the experiments, we focus on both sample complexity (defined as the number of coordinate-wise multiplications performed) and wall-clock time. In the precision-speedup tradeoff experiments, we also define the precision@k to be the proportion of the true top k algorithms returned by the algorithms' top k estimates, and define the speedup $\frac{\text{sample complexity of naïve algorithm}}{\text{sample complexity of compared algorithm}}$. In some experiments, we also allow for algorithms to identify an ϵ-suboptimal atom, where ϵ is additive error.

BanditMIPS scales as $O(1)$ with respect to d: We first assess the scaling with d for BanditMIPS on the four real-world datasets. Figure 1 shows that BanditMIPS does not scale with d and returns the correct answer to the MIPS problem. This validates our theoretical results in Section 4.

BanditMIPS exhibits lower sample complexity than baselines while preserving accuracy: We also compare the sample complexities of BanditMIPS and BanditMIPS-α to those of the eight baseline MIPS algorithms for different values of d. Due to prohibitely slow baseline algorithms for larger values of d, we omitted the `Sift-1M` and `CryptoPairs` from the real-world datasets and instead measure sample complexities on the two synthetic datasets, with a maximum

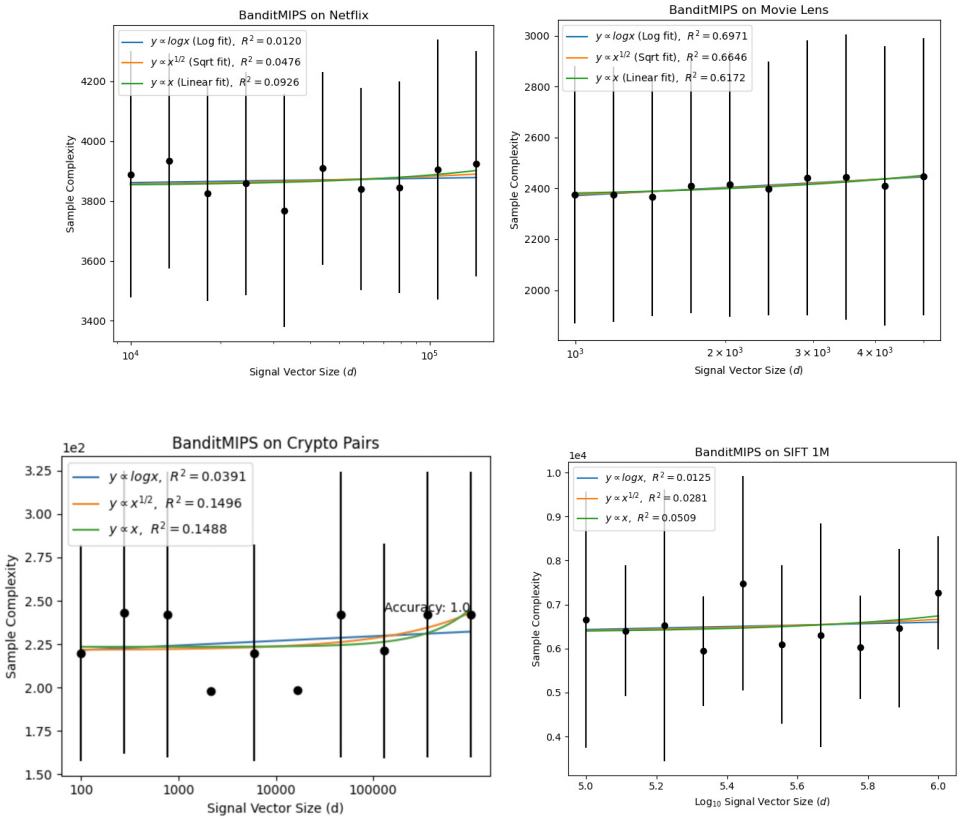


Figure 1. Sample complexity of BanditMIPS versus d on four real-world datasets. The values of R^2, the coefficient of determination, are similar for linear, logarithmic, and square root fits, which suggests the scaling is actually constant; the sample complexity of BanditMIPS does not scale with d. 95%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

of $d = 20,000$. 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 and neglect the cost of preprocessing for the baseline algorithms; this is favorable to the baselines.

Figure 2 shows that BanditMIPS and BanditMIPS-α substantially outperform other algorithms in sample complexity on all four datasets. Note that 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. In all experiments, BanditMIPS and BanditMIPS-α returned the correct answers to the MIPS problems.

BanditMIPS is faster in wall-clock time than baselines while preserving accuracy: We also measure the wall-clock performance of BanditMIPS and BanditMIPS-α compared to the naïve algorithm and the BoundedME algorithm (the closest baseline). BanditMIPS and BanditMIPS-α significantly outperform both baselines. As shown in Table 2 and Table 3, BanditMIPS surpasses BoundedME by a large margin for the Netflix and Movie Lens datasets.

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

Table 2. Wall-clock time speedups of different algorithms on the Netflix dataset. BanditMIPS and BanditMIPS-α significantly outperform the other algorithms. In this setting, $\epsilon = 0.1$, $\delta = 0.1$, $n = 1,000$, and $d = 100,000$. Confidence intervals are omitted for clarity.

BanditMIPS exhibits a better trade-off between speed and accuracy than baselines: We evaluate the trade-off between speed and accuracy of BanditMIPS and BanditMIPS-

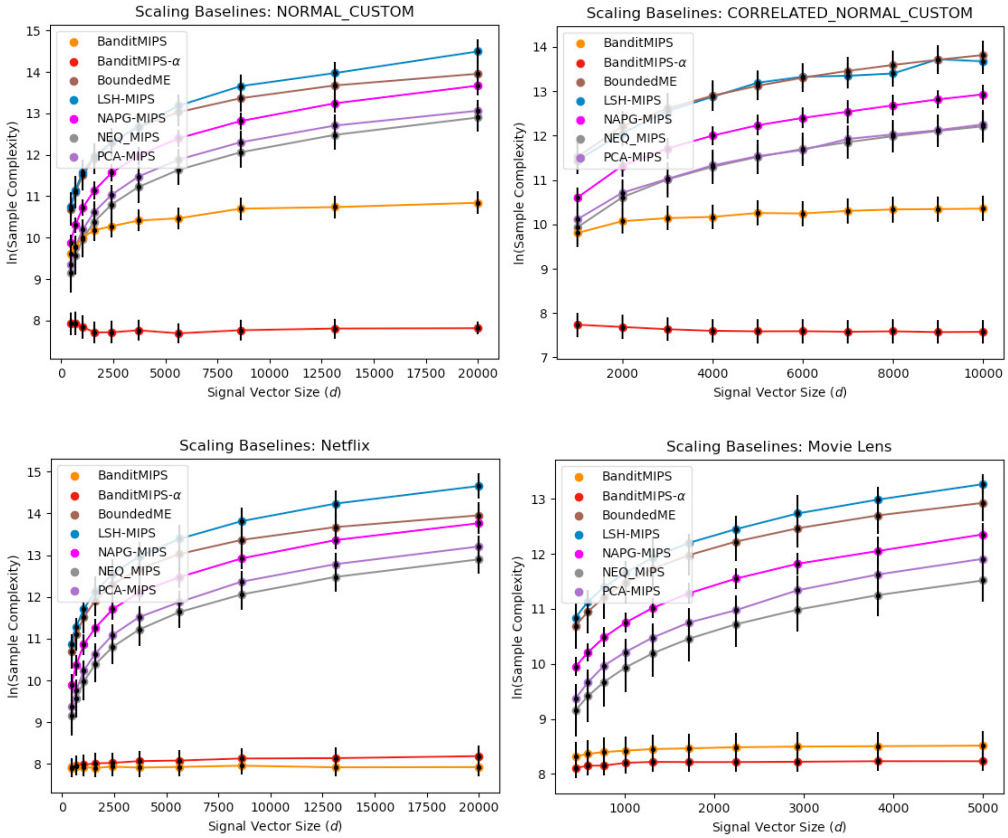
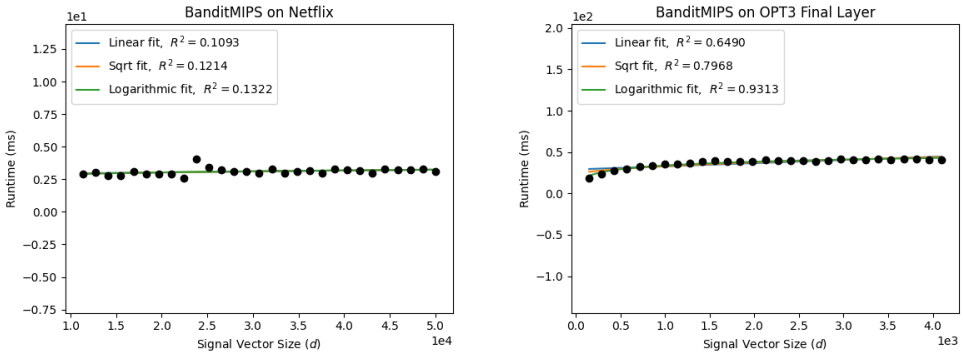


Figure 2. Sample complexities of BanditMIPS, BanditMIPS-α, and other baseline algorithms versus d across four datasets. The y-axis is on a logarithmic scale. BanditMIPS and BanditMIPS-α outperform other baselines in sample complexity. On the Movie Lens dataset, BanditMIPS outperforms the next best algorithm by $30\times$. 95% CIs are provided around the mean and are computed from 10 random trials.



(a) (b)

Figure 3. Wall-clock time of BanditMIPS versus d on the Netflix dataset and the OPT-6.7B experiment. The runtime of BanditMIPS does not scale with d. For the Netflix dataset, $\epsilon = 0.1$, $\delta = 0.1$, and $n = 1,000$. In the OPT-6.7B experiment, $\epsilon = 1.0$, $\delta = 0.9$, and $n = 1,000$. Means were calculated from 10 random seeds; confidence intervals are omitted for clarity.

α by varying the error probability δ; similarly, we vary the corresponding hyperparameters in the baseline algorithms (see Appendix B.3 for more details). As in Liu et al. (2019), we define the speedup of an algorithm to be 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

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

Table 3. Wall-clock time speedups of different algorithms on the Movie Lens dataset. BanditMIPS and BanditMIPS-α significantly outperform the other algorithms. In this setting, $\epsilon = 0.1$, $\delta = 0.1$, $n = 1,000$, and $d = 6,000$. Confidence intervals are omitted for clarity.

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

Table 4. Wall-clock time speedups of different algorithms when applied to the classification layer of OPT-6.7B. BanditMIPS and BanditMIPS-α significantly outperform the other algorithms. In this setting, $\epsilon = 1.0$, $\delta = 0.9$, $n = 10,000$, and $d = 4,096$. Confidence intervals are omitted for clarity.

returns the true MIPS solution (precision@1). Figure 4 shows the results of the precision-speedup tradeoff experiments on the Netflix and Movie Lens datasets. BanditMIPS and BanditMIPS-α demonstrate the best precision-speedup tradeoffs amongst all algorithms. Figure 4 also includes the k-MIPS setting where the goal is to find the top k atoms, with $k = 5$.

BanditMIPS can be applied in conjunction with preprocessing techniques to reduce scaling with n: Many forms of preprocessing for the MIPS problem do not affect the applicability of BanditMIPS ; in fact, BanditMIPS can often be used in conjunction with preprocessing to reduce its scaling with n. We propose a variant of BanditMIPS , dubbed Bucket_AE, that combines BanditMIPS with a normalized binning technique. More precisely, we estimate the norm of each atom with a constant number of samples. Afterwards, we sort use the results to sort the atoms into bins of b atoms in decreasing order, where b is a hyperparameter. Then, when running BanditMIPS, we make comparisons between only the best atoms in each bin and eliminate an entire bin if the maximum possible inner product 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. Figure 5 demonstrate the efficacy Bucket_AE; it reduces the scaling with n while maintaining $O(1)$ scaling with d on the Netflix dataset. In all experiments, Bucket_AE returned the correct solution to the MIPS problem for all trials.

BanditMIPS can be applied to Matching Pursuit: In the Matching Pursuit (MP) problem, 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 reiterating this process with the residual. This approach solves the MIPS problem several times as a subroutine. Our previous experimental results suggest that BanditMIPS can also be used to accelerate the MP problem.

We construct a simple synthetic dataset (titled the `SimpleSong`) where the query is constructed as five notes; each note is the best atom from an independent MIPS iteration. Figure 5 demonstrates the total sample complexity of BanditMIPS to identify these notes (five iterations of MIPS) of the song as the song length increases by looping. BanditMIPS demonstrates no scaling with respect to d as the song length increases. This experiment is described in greater detail in Appendix C.

BanditMIPS can be in classification layers of LLMs: The final operation in an LLM's inference pass in classification tasks, e.g., next-token prediction, is usually a matrix multiplication followed by a non-linearity. In such settings, where the outputs correspond to probabilities over possible next tokens, it is often sufficient to find the highest element that results from the final matrix multiplication; this is exactly the MIPS problem. We employ BanditMIPS on the classification layer of OPT-6.7B (Zhang et al., 2022), where the MIPS problem is used determine the next token to generate. Table 4 demonstrates the $4\times$ speedup achieved by BanditMIPS for hidden dimension size $d = 4,096$. Additionally, Figure 3 demonstrates that BanditMIPS still scales as $O(1)$ with respect to d in wall-clock time.

Additional Experiments and Violation of Distributional Assumptions: We discuss a dataset on which the assumptions in Section 4 fail, namely when Δ scales with d in Appendix C. We also discuss the robustness of BanditMIPS to corruption of the underlying dataset in Appendix C.

6. Conclusions and Limitations

In this work, we proposed BanditMIPS, a novel state-of-the-art algorithm for MIPS in high-dimensional settings. We also proposed BanditMIPS-α, which provides additional runtime speedup by sampling coordinates intelligently. We demonstrated, both theoretically and experimentally, that BanditMIPS and BanditMIPSachievee $O(1)$ sample complexity with respect to d under general settings. We also showed that both algorithms outperforms baseline algorithms across a variety of experimental settings and achieve up to a $30\times$ wall-clock time improvement over the next fastest algorithm. We discussed how BanditMIPS can be used as a subroutine in other applications, such as Matching

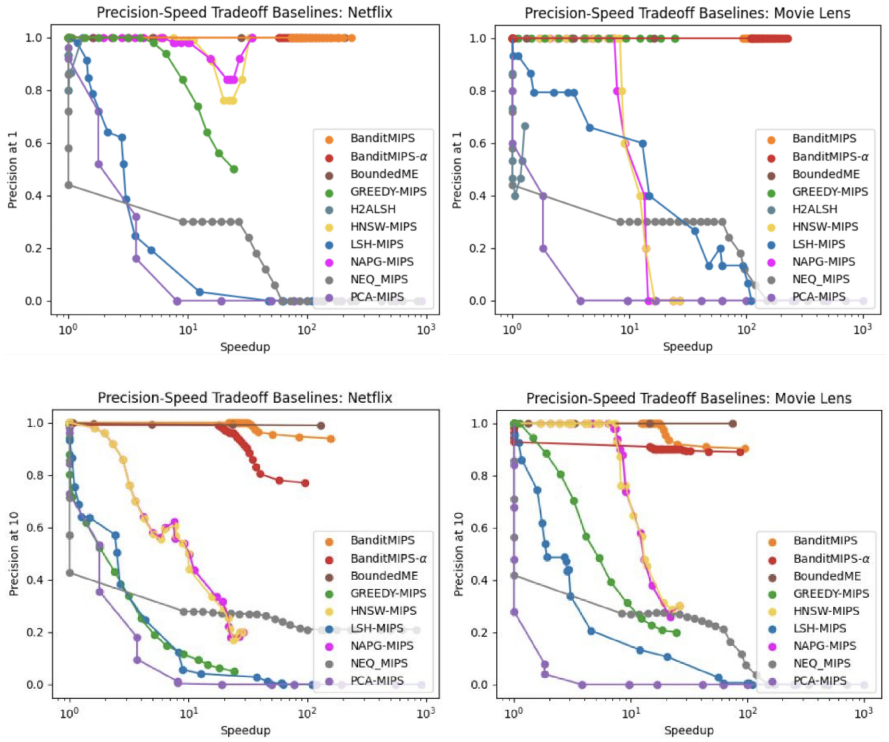


Figure 4. Precision@k versus speedup for various algorithms across different real-world datasets. Higher is better. The top row consists of experiments for precision@1 (accuracy); the bottom row for precision@10. BanditMIPS and BanditMIPS-α significantly outperform the the algorithms; they consistently achieve better precision at higher speedup values than the baselines. Each dot represents the mean across 10 random trials and CIs are omitted for clarity.

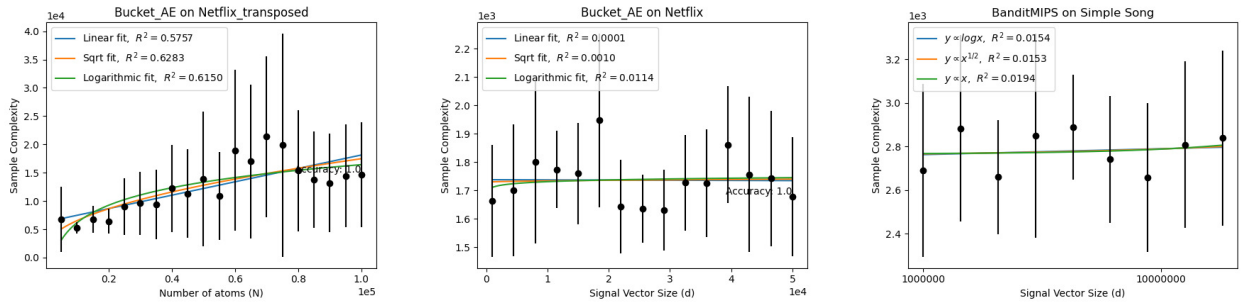


Figure 5. Sample complexity of Bucket_AE versus n (left) and d (middle) on the Netflix dataset. Bucket_AE demonstrates that no scaling with d and better scaling with n than BanditMIPS. Means and CIs were obtained from 5 random seeds. Right: sample complexity of MP when using BanditMIPS as a subroutine for MIPS versus d on the `SimpleSong` dataset. The sample complexity 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.

Pursuit and in the classification layer of a large language model (LLM), and can be used in conjunction with preprocessing techniques.

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. Furthermore, when the sub-Gaussianity parameter σ is large, BanditMIPS may collapse to the naïve algorithm.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgments

MT is supported by a Stanford Data Science Fellowship and a Stanford Interdisciplinary Graduate Fellowship.

IS was supported in part by the National Science Foundation under grant CCF-2046991.

MJZ is partially supported by the Curci Foundation Research Grant.

References

Abuzaid, F., Sethi, G., Bailis, P., and Zaharia, M. To Index or Not to Index: Optimizing Exact Maximum Inner Product Search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1250–1261, April 2019. doi: 10.1109/ICDE.2019.00114. ISSN: 2375-026X.

Amagata, D. and Hara, T. Reverse Maximum Inner Product Search: How to efficiently find users who would like to buy my item? *Fifteenth ACM Conference on Recommender Systems*, pp. 273–281, September 2021. doi: 10.1145/3460231.3474229. Conference Name: RecSys '21: Fifteenth ACM Conference on Recommender Systems ISBN: 9781450384582 Place: Amsterdam Netherlands Publisher: ACM.

Amato, G. and Savino, P. 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*, pp. 28. ICST / ACM, 2008. doi: 10.4108/ICST.INFOSCALE2008.3486.

Aouali, I., Benhalloum, A., Bompaire, M., Ait Sidi Hammou, A., Ivanov, S., Heymann, B., Rohde, D., Sakhi, O., Vasile, F., and Vono, M. 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, pp. 4772–4773, New York, NY, USA, August 2022. Association for Computing Machinery. ISBN 978-1-4503-9385-0. doi: 10.1145/3534678.3542622.

Audibert, J.-y., Bubeck, S., and Munos, R. Best arm identification in multiarmed bandits. In *In 23rd Annual Conference on Learning Theory*, 2010.

Bachrach, Y., Finkelstein, Y., Gilad-Bachrach, R., Katzir, L., Koenigstein, N., Nice, N., and Paquet, U. 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, pp. 257–264, New York, NY, USA, October 2014. Association for Computing Machinery. ISBN 978-1-4503-2668-1. doi: 10.1145/2645710.2645741.

Bagaria, V., Kamath, G. M., Ntranos, V., Zhang, M. J., and Tse, D. Medoids in almost-linear time via multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pp. 500–509, 2018a.

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

Bagaria, V., Baharav, T. Z., Kamath, G. M., and David, N. T. Bandit-based monte carlo optimization for nearest neighbors. *IEEE Journal on Selected Areas in Information Theory*, 2(2):599–610, 2021.

Baharav, T. Z. and Tse, D. Ultra fast medoid identification via correlated sequential halving. In *Advances in Neural Information Processing Systems*, pp. 3650–3659, 2019.

Baharav, T. Z., Cheng, G., Pilanci, M., and Tse, D. Approximate function evaluation via multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pp. 108–135. PMLR, 2022.

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

Bernhardsson, E. *Annoy: Approximate Nearest Neighbors in C++/Python*, 2018. URL https://pypi.org/project/annoy/. 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. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 2206–2240. PMLR, June 2022.

Boytsov, L. and Naidan, B. 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*, pp. 280–293. Springer, 2013a. doi: 10.1007/978-3-642-41062-8_28.

Boytsov, L. and Naidan, B. 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*, pp. 1574–1582, 2013b.

Boytsov, L., Novak, D., Malkov, Y. A., and Nyberg, E. 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*, pp. 1099–1108. ACM, 2016. doi: 10.1145/2983323.2983815.

Brigham, E. O. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

Carsten. 400+ crypto currency pairs at 1-minute resolution, 2022.

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

Chávez, E., Figueroa, K., and Navarro, G. Effective proximity retrieval by ordering permutations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1647–1658, 2008. doi: 10.1109/TPAMI.2007.70815.

Dai, X., Yan, X., Ng, K. K. W., Liu, J., and Cheng, J. Norm-Explicit Quantization: Improving Vector Quantization for Maximum Inner Product Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 51–58, April 2020. doi: 10.1609/aaai.v34i01.5333. ISSN: 2374-3468, 2159-5399 Issue: 01 Journal Abbreviation: AAAI.

Dasgupta, S. and Freund, Y. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 537–546, 2008.

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

Dong, W., Wang, Z., Charikar, M., and Li, K. 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*, pp. 1. ACM, 2012. doi: 10.1145/2324796.2324798.

Even-Dar, E., Mannor, S., and Mansour, Y. Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. *The Journal of Machine Learning Research*, 7:1079–1105, December 2006. ISSN 1532-4435.

Even-Dar, E., Mannor, S., and Mansour, Y. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. In *Journal of Machine Learning Research*, volume 7, pp. 1079–1105, 2006.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5 (4), dec 2015. ISSN 2160-6455. doi: 10.1145/2827872.

Hirata, K., Amagata, D., Fujita, S., and Hara, T. Solving Diversity-Aware Maximum Inner Product Search Efficiently and Effectively. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, pp. 198–207, New York, NY, USA, September 2022. Association for Computing Machinery. ISBN 978-1-4503-9278-5. doi: 10.1145/3523227.3546779.

Huang, Q., Ma, G., Feng, J., Fang, Q., and Tung, A. K. H. 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, pp. 1561–1570, New York, NY, USA, July 2018. Association for Computing Machinery. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3219971.

Jamieson, K. and Nowak, R. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, March 2014. doi: 10.1109/CISS.2014.6814096.

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

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

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

Jégou, H., Douze, M., and Schmid, C. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, January 2011. ISSN 1939-3539. doi: 10.1109/TPAMI.2010.57. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

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

Li, M., Wang, H., Yang, L., Liang, Y., Shang, Z., and Wan, H. Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Systems with Applications*, 150: 113277, 2020.

Liu, J., Yan, X., Dai, X., Li, Z., Cheng, J., and Yang, M.-C. Understanding and Improving Proximity Graph Based Maximum Inner Product Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 139–146, April 2020. doi: 10.1609/aaai.v34i01.5344. ISSN: 2374-3468, 2159-5399 Issue: 01 Journal Abbreviation: AAAI.

Liu, R., Wu, T., and Mozafari, B. 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, pp. 4376–4383, Honolulu, Hawaii, USA, January 2019. AAAI Press. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33014376.

Locatello, F., Khanna, R., Tschannen, M., and Jaggi, M. A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 860–868. PMLR, April 2017.

Lorenzen, S. S. and Pham, N. Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search (Extended Abstract). In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 4789–4793, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/652.

Lu, K. and Kudo, M. AdaLSH: Adaptive LSH for Solving c-Approximate Maximum Inner Product Search Problem. *IEICE Transactions on Information and Systems*, E104.D (1):138–145, 2021. doi: 10.1587/transinf.2020EDP7132.

Lu, Z., Hu, Y., and Zeng, B. Sampling for Approximate Maximum Search in Factorized Tensor. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 2400–2406, Melbourne, Australia, August 2017. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3. doi: 10.24963/ijcai.2017/334.

Ma, C., Yu, F., Yu, Y., and Li, W. Learning Sparse Binary Code for Maximum Inner Product Search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 3308–3312, New York, NY, USA, October 2021. Association for Computing Machinery. ISBN 978-1-4503-8446-9. doi: 10.1145/3459637.3482132.

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

Maurer, A. and Pontil, M. Empirical Bernstein Bounds and Sample Variance Penalization, 2009.

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

Morozov, S. and Babenko, A. Non-metric similarity graphs for maximum inner product search. *Advances in Neural Information Processing Systems*, 31, 2018b.

Naidan, B., Boytsov, L., and Nyberg, E. Permutation search methods are efficient, yet faster search is possible. *CoRR*, abs/1506.03163, 2015.

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

Pham, N. Sublinear maximum inner product search using concomitants of extreme order statistics. *CoRR*, abs/2012.11098, 2020a.

Pham, N. 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, pp. 1339–1347, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467345.

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

Ponomarenko, A., Avrelin, N., Naidan, B., and Boytsov, L. Comparative analysis of data structures for approximate

nearest neighbor search. *Data analytics*, pp. 125–130, 2014.

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

Sivic and Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings ninth IEEE international conference on computer vision*, pp. 1470–1477. IEEE, 2003.

Song, Y., Gu, Y., Zhang, R., and Yu, G. ProMIPS: Efficient High-Dimensional c-Approximate Maximum Inner Product Search with a Lightweight Index. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 1619–1630, April 2021. doi: 10.1109/ICDE51399.2021.00143. ISSN: 2375-026X.

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

Tan, S., Xu, Z., Zhao, W., Fei, H., Zhou, Z., and Li, P. Norm Adjusted Proximity Graph for Fast Inner Product Retrieval. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 1552–1560, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467412.

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

Wu, G., Zhu, B., Li, J., Wang, Y., and Jia, Y. H2SA-ALSH: A Privacy-Preserved Indexing and Searching Schema for IoT Data Collection and Mining. *Wireless Communications and Mobile Computing*, 2022:e9990193, April 2022. ISSN 1530-8669. doi: 10.1155/2022/9990193. Publisher: Hindawi.

Xiang, L., Yan, X., Lu, L., and Tang, B. 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, pp. 1920–1924, New York, NY, USA, July 2021. Association for Computing Machinery. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462997.

Yu, H.-F., Hsieh, C.-J., Lei, Q., and Dhillon, I. S. A Greedy Approach for Budgeted Maximum Inner Product Search. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Zhang, M., Zou, J., and Tse, D. Adaptive monte carlo multiple testing via multi-armed bandits. In *International Conference on Machine Learning*, pp. 7512–7522, 2019a.

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

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A. Proofs of Theorems

In this appendix, we present the proofs of Theorems 4.1 and 4.2.

A.1. Proof of Theorem 4.1:

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_is. 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 (we note this assumption 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 incurs $O(1)$ scalar multiplication, M also corresponds to the total number of operations up to a constant factor. □

A.2. Proof of Theorem 4.2

Proof. Since each X_{iJ} is 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

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

□

B. Description of Datasets

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

B.1. 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 B.3.

B.2. Real-world datasets

Netflix Dataset: We use a subset of the data from the Netflix Prize dataset (Bennett et al., 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. More specifically, we approximate the data matrix via its 100-factor SVD decomposition. The movie vectors are used as the query vectors and atoms and d corresponds to the number of sampled users.

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

Sift-1M Dataset: The Sift-1M dataset (Jégou et al., 2011) contains features of $n = 128$ different images, where each image is an atom with $d = 1,000,000$ dimensions.

CryptoPairs Dataset: The CryptoPairs dataset (Carsten, 2022) consists of the historical trading data of more than 400 trading pairs at 1 minute resolution reaching back until the year 2013.

We note that for all these datasets, the coordinate-wise inner products are sub-Gaussian random variables. In particular, this means the assumptions of Theorem 4.1 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 B.3.

B.3. 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 $\sigma = 1$. For the Netflix and Movie Lens datasets, the sub-Gaussianity parameter was set to $\sigma = 25$. For the `CryptoPairs`, `SIFT-1M`, and `SimpleSong` datasets described in Appendix D, the sub-Gaussianity parameters were set to $\sigma = 2.5e9$, $\sigma = 6.25e5$, and $\sigma = 25$, 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 precision versus speedup 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.

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 metrics for our results (except for wall-clock time measurements).

C. Additional Experimental Results

Assumptions on Δ: We also investigate the performance of BanditMIPS on a dataset on which the necessary distributional assumptions are violated. We call this dataset the `SymmetricNormal` dataset. In this dataset, the signal and each atom's coordinate is drawn i.i.d. from $\mathcal{N}(0, 1)$, making all atoms symmetric *a priori*. We now consider the quantity $\Delta_{i,j}(d) \coloneqq \mu_1(d) - \mu_2(d) = \frac{\mathbf{v}_1^T q - \mathbf{v}_2^T q}{d}$, i.e., the gap between the first and second arm, where our notation emphasizes we are studying each quantity as d increases. 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 4.1 then predicts that BanditMIPS scales linearly with d. 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

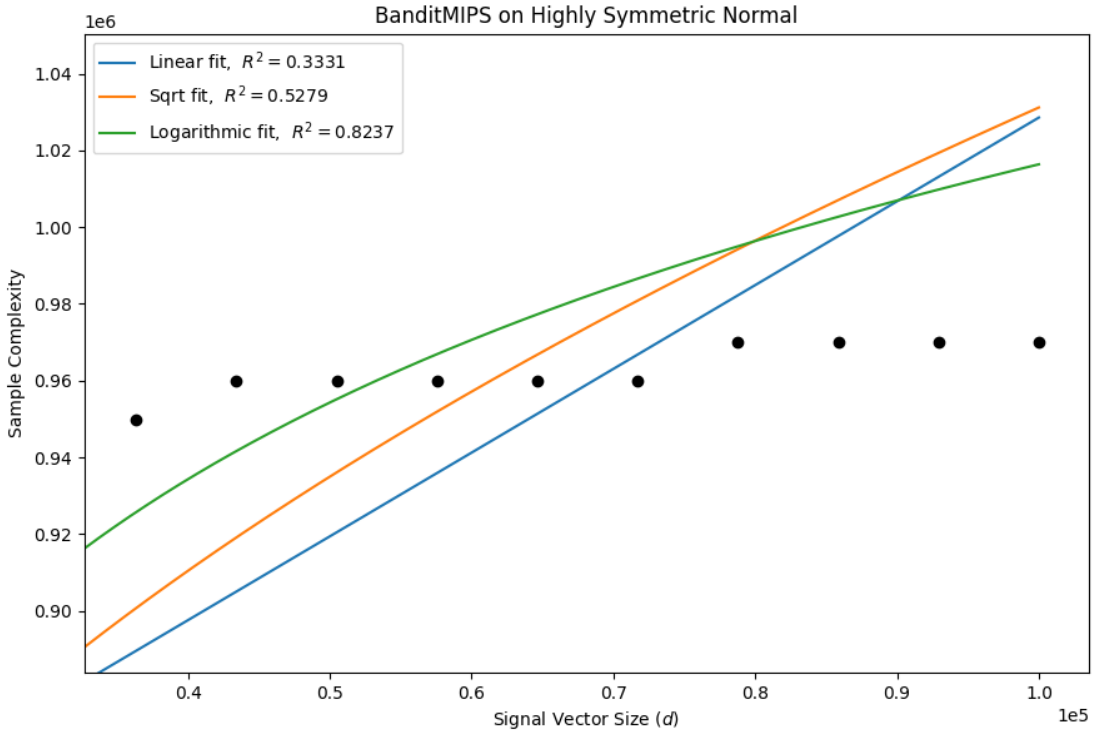


Figure 1. Sample Complexity of BanditMIPS versus d on the `SymmetricNormal` dataset when ϵ-suboptimal atoms are identified, with $\epsilon = 0.1$. BanditMIPS, with this modification, scales as $O(1)$ with respect to d, even when all atoms have an equal inner product with the query vector.

the Δ_is for large d, and instead on the relative error hyperparameter ϵ. Figure C demonstrates that, with this modification, BanditMIPS does not scale with d even in this highly symmetric dataset.

D. 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.

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

Table 5. Frequencies for various musical notes.

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

able to recover the notes played in the song in order of decreasing amplitude: G4, C5, E4, E5, and C4 in each experiment. 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.

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.