Quantization based Fast Inner Product Search

Ruiqi Guo **Sanjiv Kumar** **Krzysztof Choromanski** **David Simcha**
Google Research, New York, NY 10011, USA

Abstract

We propose a quantization based approach for fast approximate Maximum Inner Product Search (MIPS). Each database vector is quantized in multiple subspaces via a set of codebooks, learned directly by minimizing the inner product quantization error. Then, the inner product of a query to a database vector is approximated as the sum of inner products with the subspace quantizers. Different from recently proposed LSH approaches to MIPS, the database vectors and queries do not need to be augmented in a higher dimensional feature space. We also provide a theoretical analysis of the proposed approach, consisting of the concentration results under mild assumptions. Furthermore, if a small set of *held-out* samples from the query distribution is given at the training time, we propose a modified codebook learning procedure which further improves the accuracy. Experimental results on a variety of datasets including those arising from deep neural networks show that the proposed approach significantly outperforms the existing state-of-the-art.

1 Introduction

Many information processing tasks such as retrieval and classification involve computing the inner product of a query vector with a set of database vectors, with the goal of returning the database instances having the largest inner products. This is often called Maximum Inner Product Search (MIPS) problem. Formally, given a database $X = \{x_i\}_{i=1\cdots n}$, and a query vector q drawn from the query distribution, where $x_i, q \in \mathbb{R}^d$, we want to find $x_q^* \in X$ such that $x_q^* = \text{argmax}_{x\in X}(q^T x)$. This definition can be trivially extended to return top-N largest inner products.

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41. Copyright 2016 by the authors.

The MIPS problem is particularly appealing for large scale applications. For example, a recommendation system needs to retrieve the most relevant items to a user from an inventory of millions of items, whose relevance is commonly represented as inner products [7]. Similarly, a large scale classification system needs to classify an item into one of the categories, where the number of categories may be very large [9]. A brute-force computation of inner products via a linear scan requires $O(nd)$ time and space, which becomes computationally prohibitive when the number of database vectors and the data dimensionality is large. Therefore it is valuable to consider algorithms that can compress the database X and compute approximate x_q^* much faster than the brute-force search.

The problem of MIPS is related to that of Nearest Neighbor Search with respect to L_2 distance (L_2NNS) or angular distance (θNNS) between a query and a database vector:

$$q^T x = 1/2(||x||^2 + ||q||^2 - ||q-x||^2) = ||q||||x|| \cos\theta,$$

or

$$\begin{aligned}\underset{x\in X}{\text{argmax}}(q^T x) &= \underset{x\in X}{\text{argmax}}(||x||^2 - ||q-x||^2)\\ &= \underset{x\in X}{\text{argmax}}(||x|| cos\theta),\end{aligned}$$

where $||.||$ is the L_2 norm. Indeed, if the database vectors are scaled such that $||x|| = \text{constant} \quad \forall x \in X$, the MIPS problem becomes equivalent to L$_2$NNS or θNNS problems, which have been studied extensively in the literature. However, when the norms of the database vectors vary, as often true in practice, the MIPS problem becomes quite challenging. The inner product (distance) does not satisfy the basic axioms of a metric such as triangle inequality and co-incidence. For instance, it is possible to have $x^T x \leq x^T y$ for some $y \neq x$. In this paper, we focus on the MIPS problem where both database and the query vectors can have arbitrary norms.

As the main contribution of this paper, we develop a Quantization-based Inner Product (QUIP) search method to address the MIPS problem. We formulate the problem of quantization as that of codebook learning, which directly minimizes the quantization error in inner products (Sec. 3). Furthermore, if a small set of *held-out* datatpoints (sampled from the query distribution but separate from testing queries) is provided at the training time, we propose a constrained optimization framework which further improves the accuracy (Sec. 3.2). We also provide a concentration-based theoretical analysis of the proposed method (Sec. 4). Extensive experiments on four real-world datasets, involving recommendation (*Movielens*, *Netflix*) and deep-learning based classification (*ImageNet* and *VideoRec*) tasks show that the proposed approach consistently outperforms the 5 state-of-the-art techniques under both fixed space and fixed time scenarios (Sec. 5).

2 Related works

The MIPS problem has been studied for more than a decade. For instance, Cohen et al. [6] studied it in the context of document clustering and presented a method based on randomized sampling without computing the full matrix-vector multiplication. In [13, 16], the authors described a procedure to modify tree-based search to adapt to MIPS criterion. Recently, Bachrach et al. [3] proposed an approach that transforms the input vectors such that the MIPS problem becomes equivalent to the L_2NNS problem in the transformed space, which they solved using a PCA-Tree.

The MIPS problem has received a renewed attention with the recent seminal work from Shrivastava and Li [18], which introduced an Asymmetric Locality Sensitive Hashing (ALSH) technique with provable search guarantees. They also transform MIPS into L_2NNS, and use the popular LSH technique [1]. Specifically, ALSH applies different vector transformations to a database vector x and the query q, respectively:

$$\hat{x} = [\tilde{x}; ||\tilde{x}||^2; ||\tilde{x}||^4; \cdots ||\tilde{x}||^{2^m}].$$
$$\hat{q} = [q; 1/2; 1/2; \cdots ; 1/2].$$

where $\tilde{x} = U_0 \frac{x}{\max_{x \in X} ||x||}$, U_0 is some constant that satisfies $0 < U_0 < 1$, and m is a nonnegative integer. Hence, x and q are mapped to a new $(d+m)$ dimensional space asymmetrically. Shrivastava and Li [18] showed that when $m \to \infty$, MIPS in the original space is equivalent to L_2NNS in the new space. The proposed hash function followed L_2LSH form [1]: $h_i^{L2}(\hat{x}) = \lfloor \frac{P_i^T \hat{x} + b_i}{r} \rfloor$,

where P_i is a $(d+m)$-dimensional vector whose entries are sampled i.i.d from the standard Gaussian, $\mathcal{N}(0,1)$, and b_i is sampled uniformly from $[0, r]$. The same authors later proposed an improved version of ALSH based on Signed Random Projection (SRP) [19]. It transforms each vector using a slightly different procedure and represents it as a binary code. Then, Hamming distance is used for MIPS.

$$\hat{x} = [\tilde{x}; \frac{1}{2} - ||\tilde{x}||^2; \frac{1}{2} - ||\tilde{x}||^4; \cdots \frac{1}{2} - ||\tilde{x}||^{2^m}],$$
$$\hat{q} = [q; 0; 0; \cdots ; 0], \quad \text{and}$$

$$h_i^{SRP}(\hat{x}) = sign(P_i^T \hat{x});$$
$$Dist^{SRP}(x, q) = \sum_{i=1}^{b} h_i^{SRP}(\hat{x}) \neq h_i^{SRP}(\hat{q}).$$

Neyshabur and Srebro [15] argued that a symmetric transformation was sufficient to develop a provable LSH approach for the MIPS problem if query was restricted to unit norm. They used a transformation similar to the one used by Bachrach et al. [3] to augment the original vectors:

$$\hat{x} = [\tilde{x}; \sqrt{1 - ||\tilde{x}||^2}]. \quad \hat{q} = [\tilde{q}; 0].$$

where $\tilde{x} = \frac{x}{max_{x \in X} ||x||}$, $\tilde{q} = \frac{q}{||q||}$. They showed that this transformation led to significantly improved results over the SRP based LSH from [19].

Recently, composite quantization [10] and additive quantization [2] techniques have been proposed to perform fast inner product search by exploiting additive nature of inner product. This problem is also related to locally learned codebook of product quantization [12] in Euclidean space. All of the above three methods learn optimized codebooks at the cost of added overhead of look up table construction, which grows linearly with the length of codes. As confirmed in [21], this overhead becomes non-negligible in practice.

In this paper, we also take a quantization based view of the MIPS problem. We differ from the above-mentioned quantization approaches in three respects: **1) Our proposed quantization scheme extends product codes to inner product search and directly optimizes the retrieval rank.** This additional constraint provides a considerable improvement over minimizing quantization error. **2) We provide theoretical guarantees.** Unlike the previous quantization based approaches, we prove unbiasedness of the estimator as well as gave concentration bounds. **3) Our lookup table construction is efficient.** Unlike the composite quantization [10] or additive quantization [2], our lookup table construction has the same speed as that of the original product quantization [11] and does not grow with the length of the code. We show our method leads to even better accuracy under both fixed space or fixed time budget on a variety of real world tasks.

3 Quantization-based inner product (QUIP) search

Instead of augmenting the input vectors to a higher dimensional space as in [15, 18], we approximate the inner products by mapping each vector to a set of subspaces, followed by independent quantization of database vectors in each subspace. In this work, we use a simple procedure for generating the subspaces. Each vector's elements are first permuted using a random (but fixed) permutation[1]. Then each permuted vector is mapped to K subspaces using simple chunking, as done in product codes [17, 11]. For ease of notation, in the rest of the paper we will assume that both query and database vectors have been permuted. Chunking leads to block-decomposition of the query $q \sim \mathbf{Q}$ and each database vector $x \in X$:

$$x = [x^{(1)}; x^{(2)}; \cdots ; x^{(K)}] \quad q = [q^{(1)}; q^{(2)}; \cdots ; q^{(K)}],$$

where each $x^{(k)}, q^{(k)} \in \mathbb{R}^l, l = \lceil d/K \rceil$.[2] The kth subspace containing the kth blocks of all the database vectors, $\{x^{(k)}\}_{i=1 \ldots n}$, is then quantized by a codebook $U^{(k)} \in \mathbb{R}^{l \times C_k}$ where C_k is the number of quantizers in subspace k. Without loss of generality, we assume $C_k = C \quad \forall\ k$. Then, each database vector x is quantized in the kth subspace as $x^{(k)} \approx U^{(k)} \alpha_x^{(k)}$, where $\alpha_x^{(k)}$ is a C-dimensional one-hot assignment vector with exactly one 1 and rest 0. Thus, a database vector x is quantized by a single dictionary element $u_x^{(k)}$ in the kth subspace.

Given the quantized database vectors, the exact inner product is approximated as:

$$q^T x = \sum_k q^{(k)T} x^{(k)} \approx \sum_k q^{(k)T} U^{(k)} \alpha_x^{(k)} = \sum_k q^{(k)T} u_x^{(k)} \tag{1}$$

Note that this approximation is 'asymmetric' in the sense that only database vectors x are quantized, not the query vector q. One can quantize q as well but it will lead to increased approximation error. In fact, the above asymmetric computation for all the database vectors can still be carried out very efficiently via look up tables similar to [11], except that each entry in the kth table is a dot product between $q^{(k)}$ and columns of $U^{(k)}$.

Before describing the learning procedure for the codebooks $U^{(k)}$ and the assignment vectors $\alpha_x^{(k)}\ \forall\ x, k$, we first show an interesting property of the approximation in (1). Let $S_c^{(k)}$ be the cth partition of the database vectors in subspace k such that $S_c^{(k)} = \{x^{(k)} : \alpha_x^{(k)}[c] = 1\}$, where $\alpha_x^{(k)}[c]$ is the cth element of $\alpha_x^{(k)}$ and $U_c^{(k)}$ is the cth column of $U^{(k)}$.

Lemma 3.1. *If* $U_c^{(k)} = \frac{1}{|S_c^{(k)}|} \sum_{x^{(k)} \in S_c^{(k)}} x^{(k)}$, *then (1) is an unbiased estimator of* $q^T x$.

Proof.

$$\begin{aligned}
&\mathop{\mathbb{E}}_{q \sim \mathbf{Q}, x \in X} [q^T x - \sum_k q^{(k)T} u_x^{(k)}] \\
&= \sum_k \mathop{\mathbb{E}}_{q \sim \mathbf{Q}} q^{(k)T} \mathop{\mathbb{E}}_{x \in X} [(x^{(k)} - u_x^{(k)}] \\
&= \sum_k \mathop{\mathbb{E}}_{q \sim \mathbf{Q}} q^{(k)T} \mathop{\mathbb{E}}_{x \in X} [\sum_c \mathbb{I}[x^{(k)} \in S_c^{(k)}](x^{(k)} - U_c^{(k)})] = 0.
\end{aligned}$$

Where $\mathbb{I}$ is the indicator function, and the last equality holds because for each k, $\mathbb{E}_{x \in S_c^{(k)}} [x^{(k)} - U_c^{(k)}] = 0$ by definition. □

We will provide the concentration inequalities for the estimator in (1) in Sec. 4. Next we describe the learning of quantization codebooks in different subspaces. We focus on two different training scenarios: when only the database vectors are given (Sec. 3.1), and when a *held-out* set of samples from the query distribution is also provided (Sec. 3.2). The latter can result in significant performance gain when queries do not follow the same distribution as the database vectors. Note that the actual queries used at the test time are different from the *held-out* samples, and hence unknown at the training time.

3.1 Learning quantization codebooks from database

Our goal is to learn data quantizers that minimize the quantization error due to the inner product approximation given in (1). Given a *held-out* set of samples Z, which is sampled from the same distribution as query $\mathbf{Q}$, but is different from testing queries, and assuming each subspace to be independent, the expected squared error can be expressed as:

$$\begin{aligned}
&\frac{1}{|Z|} \sum_{z \in Z} \sum_{x \in \mathbf{X}} [z^T x - \sum_k z^{(k)T} U^{(k)} \alpha_x^{(k)}]^2 \\
&= \sum_k \frac{1}{|Z|} \sum_{z \in Z} \sum_{x \in X} [z^{(k)T} (x^{(k)} - u_x^{(k)})]^2 \\
&= \sum_k \sum_{x \in X} (x^{(k)} - u_x^{(k)})^T \Sigma_Z^{(k)} (x^{(k)} - u_x^{(k)}),
\end{aligned} \tag{2}$$

where $\Sigma_Z^{(k)} = \frac{1}{|Z|} \sum_{z \in Z} z^{(k)} z^{(k)T}$ is the non-centered covariance matrix in subspace k estimated from *held-out* sample set Z. Minimizing the error in (2) is equivalent to solving a modified *k-Means* problem in each subspace independently. Instead of using the

[1] Another possible choice is random rotation of the vectors which is slightly more expensive than permutation but leads to improved theoretical guarantees as discussed in the supplementary material.

[2] One can do zero-padding wherever necessary, or use different dimensions in each block.

Euclidean distance, Mahalanobis distance specified by $\Sigma_Z^{(k)}$ is used for assignment. One can use the standard Lloyd's algorithm to find the solution for each subspace k iteratively by alternating between two steps:

$$c_x^{(k)} = \operatorname*{argmin}_c (x^{(k)} - U_c^{(k)})^T \Sigma_Z^{(k)} (x^{(k)} - U_c^{(k)}),$$
$$\alpha_x^{(k)}[c_x^{(k)}] = 1, \quad \forall\ c, x$$
$$U_c^{(k)} = \frac{\sum_{x^{(k)} \in S_c^{(k)}} x^{(k)}}{|S_c^{(k)}|} \quad \forall\ c. \tag{3}$$

The Lloyd's algorithm is known to converge to a local minimum (except in pathological cases where it may oscillate between equivalent solutions) [5]. Also, note that the resulting quantizers are always the Euclidean means of their corresponding partitions, and hence, Lemma 3.1 is applicable to (2) as well, leading to an unbiased estimator.

The above procedure requires the non-centered covariance matrix Σ_Z, which will not be known if *held-out* samples are not available at the training time. In that case, one possibility is to assume that the queries come from the same distribution as the database vectors, i.e., $\Sigma_Z = \Sigma_X$. In the experiments we will show that this version performs reasonably well. However, if a small set of example queries is available at the training time, besides estimating the query covariance matrix, we propose to impose novel constraints that lead to improved quantization, as described next.

3.2 Learning quantization codebook from database and *held-out* samples

In most applications, it is possible to have access to a small set of *held-out* samples from the query distribution, Z. Of course, the actual testing queries used at the test-time do not intersect with this set. Given these *held-out* samples, we propose to modify the learning criterion by imposing additional constraints while minimizing the expected quantization error. Given a *held-out* sample z, since we are interested in finding the database vector x_z^* with highest dot-product, ideally we want the dot product of query to the quantizer of x_z^* to be larger than the dot product with any other quantizers. Let us denote the matrix containing the k^{th} subspace assignment vectors $\alpha_x^{(k)}$ for all the database vectors by $A^{(k)}$. Thus, the modified optimization is given as,

$$\operatorname*{argmin}_{U^{(k)}, A^{(k)}} \sum_{z \in Z} \sum_{x \in X} [\sum_k z^{(k)T} x^{(k)} - \sum_k z^{(k)T} U^{(k)} \alpha_x^{(k)}]^2$$
$$s.t.\ \forall z, x, \quad \sum_k z^{(k)T} U^{(k)} \alpha_x^{(k)} \leq \sum_k z^{(k)T} U^{(k)} \alpha_{x_z^*}^{(k)}$$
$$\text{where} \quad x_z^* = \operatorname*{argmax}_x z^T x \tag{4}$$

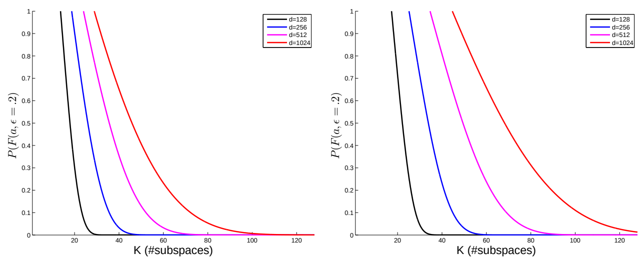
We relax the above hard constraints using slack variables to allow for some violations, which leads to the following equivalent objective:

$$\operatorname*{argmin}_{U^{(k)}, A^{(k)}} \sum_{z \in Z} \sum_{x \in X} \sum_k \left(z^{(k)T} (x^{(k)} - U^{(k)} \alpha_x^{(k)}) \right)^2$$
$$+ \lambda \sum_{z \in Z} \sum_{x \in X} [\sum_k z^{(k)T} (U^{(k)} \alpha_x^{(k)} - U^{(k)} \alpha_{x_z^*}^{(k)})]_+ \tag{5}$$

where $[t]_+ = max(t, 0)$ is the standard hinge loss, and λ is a nonnegative coefficient. We use an iterative procedure to solve the above optimization, which alternates between solving $U^{(k)}$ and $A^{(k)}$ for each k. In the beginning, each codebook $U^{(k)}$ is initialized with a set of random database vectors mapped to the k^{th} subspace. Then, we iterate through the following three steps:

1. Find a set of violated constraints W with each element as a triplet, i.e., $W_j = \{z_j, x_{z_j}^*, x_j^-\}_{j=1 \cdots J}$, where $z_j \in Z$ is an *held-out* sample, $x_{z_j}^*$ is the database vector having the maximum dot product with z_j, and z_j^- is a vector such that $z_j^T x_{z_j}^* \geq z_j^T x_j^-$ but
$$\sum_k z_j^{(k)T} U^{(k)} \alpha_{x_{z_j}^*}^{(k)} < \sum_k z_j^{(k)T} U^{(k)} \alpha_{x_j^-}^{(k)}$$
2. Fixing $U^{(k)}$ and all columns of $A^{(k)}$ except $\alpha_x^{(k)}$, one can update $\alpha_x^{(k)}\ \forall\ x, k$ as:
$$c_x^{(k)} = \operatorname*{argmin}_c \big((x^{(k)} - U_c^{(k)})^T \Sigma_Z^{(k)} (x^{(k)} - U_c^{(k)})$$
$$+ \lambda \big(\sum_j z^{(k)T} U_c^{(k)} (\mathbb{I}[x = x_j^-] - \mathbb{I}[x = x_{z_j}^*]) \big) \big),$$
$$\alpha_x^{(k)}[c_x^{(k)}] = 1$$
Since C is typically small (256 in our experiments), we can find $c_x^{(k)}$ by enumerating all possible values of c.
3. Fixing A, and all the columns of $U^{(k)}$ except $U_c^{(k)}$, one can update $U_c^{(k)}$ by gradient descent where gradient can be computed as:
$$\nabla U_c^{(k)} = 2\Sigma_Z^{(k)} \sum_{x \in X} \alpha_x^{(k)}[c] (U_c^{(k)} - x^{(k)})$$
$$+ \lambda \sum_j \left(z_j^{(k)} (\alpha_{x_j^-}^{(k)}[c] - \alpha_{x_{z_j}^*}^{(k)}[c]) \right)$$

Note that if no violated constraint is found, step 2 is equivalent to finding the nearest neighbor of $x^{(k)}$ in $U^{(k)}$in Mahalanobis space specified by $\Sigma_Z^{(k)}$. Also, in that case, by setting $\nabla U_c^{(k)} = 0$, the update rule in step 3 becomes $U_c^{(k)} = \frac{1}{|S_c^{(k)}|} \sum_{x^{(k)} \in S_c^{(k)}} x^{(k)}$ which



(a) $\eta = 0.75$ (b) $\eta = 0.5$

Figure 1: Upper bound on the probability of an event $\mathcal{F}(a,\epsilon)$ as a function of the number of subspaces K for $\epsilon = 0.2$. The left figure corresponds to $\eta = 0.75$ and the right one to $\eta = 0.5$. Different curves correspond to different data dimensionality ($d = 128, 256, 512, 1024$).

is the stationary point for the first term. Thus, if no constraints are violated, the above procedure becomes identical to *k-Means*-like procedure described in Sec. 3.1. The steps 2 and 3 are guaranteed not to increase the value of the objective in (4). In practice, we have found that the iterative procedure can be significantly sped up by modifying the step 3 as perturbation of the stationary point of the first term with a single gradient step of the second term. The time complexity of step 1 is at most $O(nKC|Z|)$, but in practice it is much cheaper because we limit the number of constraints in each iteration to be at most J. Step 2 takes $O(nKC)$ and step 3 $O((n+J)KC)$ time. In all the experiments, we use at most $J = 1000$ constraints in each iteration, Also, we fix $\lambda = .01$, step size $\eta_t = 1/(1+t)$ at each iteration t, and the maximum number of iterations $T = 30$.

4 Theoretical analysis

In this section we present concentration results about the quality of the quantization-based inner product search method. Due to the space constraints, proofs of the theorems are provided in the supplementary material. We start by defining a few quantities.

Definition 4.1. *Given fixed $a, \epsilon > 0$, let $\mathcal{F}(a,\epsilon)$ be an event such that the exact dot product $q^T x$ is at least a, but the quantized version is either smaller than $q^T x(1-\epsilon)$ or larger than $q^T x(1+\epsilon)$.*

Intuitively, the probability of event $\mathcal{F}(a,\epsilon)$ measures the chance that difference between the exact and the quantized dot product is large, when the exact dot product is large. We would like this probability to be small. Next, we introduce the concept of balancedness for subspaces.

Definition 4.2. *Let v be a vector which is chunked into K subspaces: $v^{(1)}, ..., v^{(K)}$. We say that chunking is η-balanced if the following holds for every $k \in \{1, ..., K\}$:*

$$\|v^{(k)}\|^2 \le (\frac{1}{K} + (1-\eta))\|v\|^2$$

Since the input data may not satisfy the balancedness condition, we next show that random permutation tends to create more balanced subspaces. Obviously, a (fixed) random permutation applied to vector entries does not change the dot product.

Theorem 4.1. *Let v be a vector of dimensionality d and let $perm(v)$ be its version after applying random permutation of its dimensions. Then the expected $perm(v)$ is 1-balanced.*

Another choice of creating balancedness is via a (fixed) random rotation, which also does not change the dot-product. This leads to even better balancedness property as discussed in the supplementary material (see Theorem 2.1). Next we show that the probability of $\mathcal{F}(a,\epsilon)$ can be upper bounded by an exponentially small quantity in K, indicating that the quantized dot products accurately approximate large exact dot products when the quantizers are the means obtained from Mahalanobis *k-Means* as described in Sec. 3.1. Note that in this case quantized dot-product is an unbiased estimator of the exact dot-product as shown in Lemma 3.1.

Theorem 4.2. *Assume that the dataset X of dimensionality d resides entirely in the ball $\mathcal{B}(p,r)$ of radius r, centered at p . Further, let $\{x - p : x \in X \backslash p\}$ be η-balanced for some $0 < \eta < 1$, where $\backslash$ is applied pointwise, and let $\mathbb{E}[\sum_k (x^{(k)} - u_x^{(k)})]_{k=1 \cdots K}$ be a martingale. Denote $q_{max} = \max_{k=1,...,K} \max_{q \in Q} \|q^{(k)}\|$. Then, there exist K sets of codebooks, each with C quantizers, such that the following is true:*

$$\mathbb{P}(\mathcal{F}(a,\epsilon)) \le 2e^{-(\frac{a\epsilon}{r})^2 \frac{C^{\frac{2K}{d}}}{8q_{max}^2(1+(1-\eta)K)}}.$$

The above theorem shows that the probability of $\mathcal{F}(a,\epsilon)$ decreases exponentially as the number of subspaces (i.e., blocks) K increases. This is consistent with experimental observation that increasing K leads to more accurate retrieval. Examples of how the upper bound of $\mathcal{F}(a,\epsilon)$ decreases is illustrated in Figure 1.

Furthermore, if we assume that each subspace is independent, which is a slightly more restrictive assumption than the martingale assumption made in Theorem 4.2, we can use Berry-Esseen [14] inequality to obtain an even stronger upper bound as given below.

Theorem 4.3. *Suppose, $\Delta = \max_{k=1,...,K} \Delta^{(k)}$, where $\Delta^{(k)} = \max_x ||u_x^{(k)} - x^{(k)}||$ is the maximum distance between a datapoint and its quantizer in subspace k. Assume $\Delta \le \frac{a^{\frac{1}{3}}}{q_{max}}$. Then,*

$$\mathbb{P}(\mathcal{F}(a,\epsilon)) \le \frac{2\sum_{k=1}^K L^{(k)}}{\sqrt{2\pi}|X|a\epsilon} e^{-\frac{a^2\epsilon^2|X|^2}{2(\sum_{k=1}^K L^{(k)})^2}} + \frac{\beta K(\sum_{k=1}^K L^{(k)})^{\frac{3}{2}}}{a^2\epsilon^3|X|^{\frac{3}{2}}},$$

where $L^{(k)} = E_{q\in Q}[\sum_{S_c^{(k)}}\sum_{x\in S_c^{(k)}}(q^{(k)T}x^k - q^{(k)T}u_x^{(k)})^2]$ *and* $\beta > 0$ *is some universal constant.*

5 Experimental results

We conducted experiments with 4 datasets which are summarized below:

Movielens This dataset consists of user ratings collected by the MovieLens site from web users. We use the same SVD setup as described in the ALSH paper [18] and extract 150 latent dimensions from SVD results. This dataset contains 10,681 database vectors and 71,567 query vectors.

Netflix The Netflix dataset comes from the Netflix Prize challenge [4]. It contains 100,480,507 ratings that users gave to Netflix movies. We process it in the same way as suggested by [18]. That leads to 300 dimensional data. There are 17,770 database vectors and 480,189 query vectors.

ImageNet This dataset comes from the state-of-the-art GoogLeNet [20] image classifier trained on ImageNet[3]. The goal is to speed up the maximum dot-product search in the last i.e., classification layer. Thus, the weight vectors for different categories form the database while the query vectors are the last hidden layer embeddings from the ImageNet validation set. The data has 1025 dimensions (1024 weights and 1 bias term). There are 1,000 database and 49,999 query vectors.

VideoRec This dataset consists of embeddings of user interests [8], trained via a deep neural network to predict a set of relevant videos for a user. The number of videos in the repository is 500,000. The network is trained with a multi-label logistic loss. As for the *ImageNet* dataset, the last hidden layer embedding of the network is used as query vector, and the classification layer weights are used as database vectors. The goal is to speed up the maximum dot product search between a query and 500,000 database vectors. Each database vector has 501 dimensions (500 weights and 1 bias term). The query set contains 1,000 vectors.

Following [18], we focus on retrieving Top-1, 5 and 10 highest inner product neighbors for Movielens and Netflix experiments. For ImageNet dataset, we retrieve top-5 categories as common in the literature. For the VideoRec dataset, we retrieve Top-50 videos for recommendation to a user. We experiment with three variants our technique: (1) ***QUIP-cov(x)***: uses only database vectors at training, and replaces $\Sigma_{\mathbf{Z}}$ by Σ_X in the *k-Means* like codebook learning in Sec. 3.1, (2) ***QUIP-cov(z)***: uses Σ_Z estimated from a *held-out* sample set for *k-Means* like codebook learning, and (3) ***QUIP-opt***: uses full optimization based quantization (Sec. 3.2). We compare the performance (precision-recall curves) with 5 state-of-the-art methods: (1) ***Signed ALSH*** [18], (2) ***L2 ALSH*** [18][4]; (3) ***Simple LSH*** [15]. (4) PCA-tree version adapted to inner product search as proposed in [3], which has shown better results than IP-tree [16] and (5) Composite quantization [10] which also uses quantization view and codebook learning.

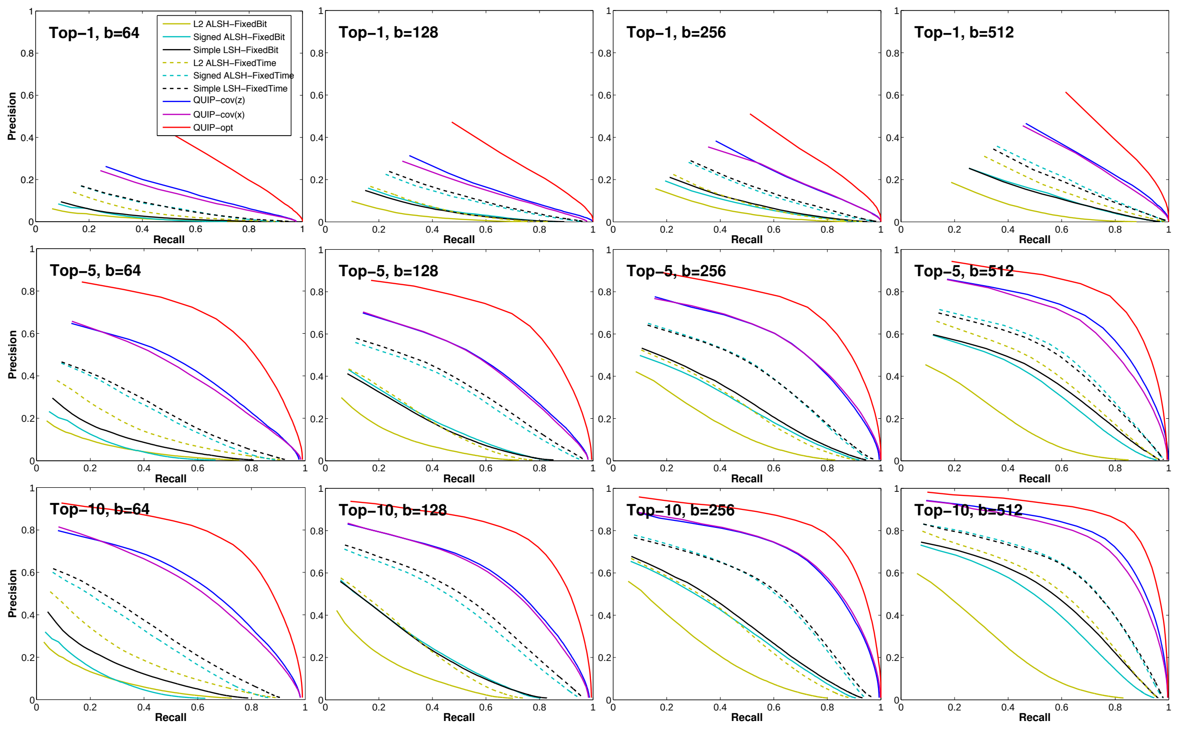
We conduct two sets of experiments: (i) *fixed bit* - the number of bits used by all the techniques is kept the same, (ii) *fixed time* - the time taken by all the techniques is fixed to be the same. In the fixed bit experiments, we fix the number of bits to be $b = 64, 128, 256, 512$. For all the *QUIP* variants, the codebook size for each subspace, C, was fixed to be 256, leading to a 8-bit representation of a database vector in each subspace. The number of subspaces (i.e., blocks) was varied to be $k = 8, 16, 32, 64$ leading to $64, 128, 256, 512$ bit representation, respectively. For the fixed time experiments, we first note that the proposed *QUIP* variants use table lookup based distance computation while the LSH based techniques use POPCNT-based Hamming distance computation. Depending on the number of bits used, we found POPCNT to be 2 to 3 times faster than table lookup. Thus, in the fixed-time experiments, we increase the number of bits for LSH-based techniques by 3 times to ensure that the time taken by all the methods is the same.

Figure 2 shows the precision recall curves for *Movielens* and *Netflix*, and Figure 3 shows the same for the *ImageNet* and *VideoRec* datasets. All the quantization based approaches outperform LSH based methods significantly when all the techniques use the same number of bits. Even in the fixed time experiments, the proposed approaches remain superior to the LSH-based approaches (shown with dashed curves), even though the former uses 3 times less bits than latter, leading to significant reduction in memory footprint. We also show comparison with *Composite Quantization* [10] and *PCA-Tree* [3] in Figure 4. *Composite Quantization* [10] works better than ALSH and SimpleLSH with the same number of bits, but is outperformed by our proposed methods. *PCA-Tree* does not perform well on these datasets, mostly due to the fact that the dimensionality of our datasets is relatively high (150 to 1025 dimensions), and trees are known to be more susceptible to dimensionality. Note the the original paper from Bachrach et al. [3] used datasets with dimensionality up to 50.

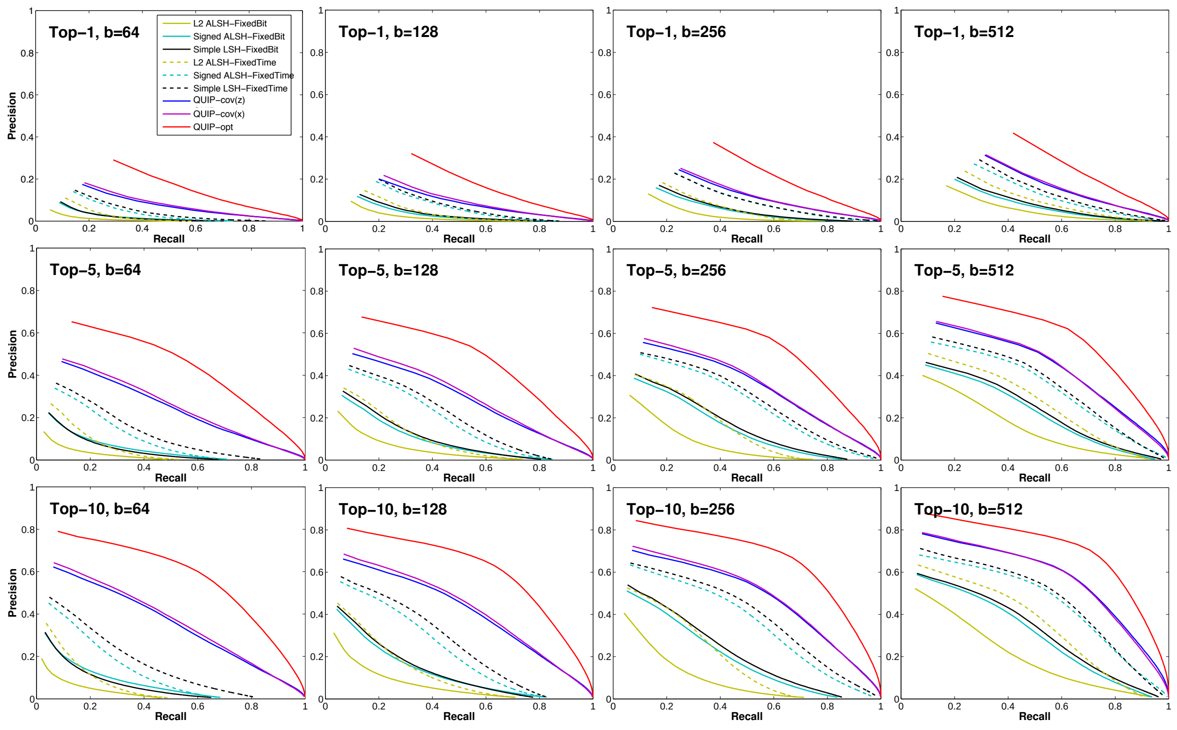
Among the proposed methods, *QUIP-cov(z)* typically performs better than *QUIP-cov(x)*, but the gap in per-

[3]The original paper ensembled 7 models and used 144 different crops. In our experiment, we focus on one global crop using one model.

[4]The recommended parameters $m = 3, U_0 = 0.85, r = 2.5$ were used in the implementation.

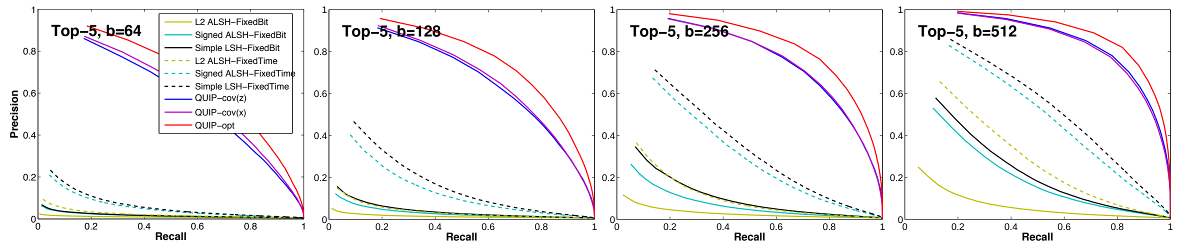


(a) Movielens dataset

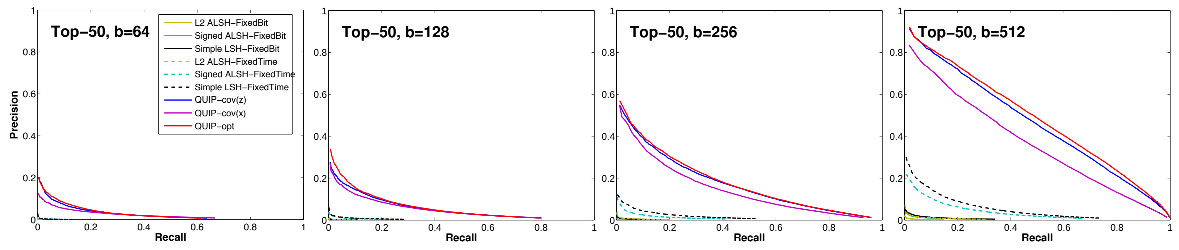


(b) Netflix dataset

Figure 2: Precision Recall curves (higher is better) for different methods on Movielens and Netflix datasets, retrieving Top-1, 5 and 10 items. **Baselines:** *Signed ALSH* [19], *L2 ALSH* [18] and *Simple LSH* [15]. **Proposed Methods:** *QUIP-cov(x)*, *QUIP-cov(z)*, *QUIP-opt*. Curves for fixed bit experiments are plotted in solid line for both the baselines and proposed methods, where the number of bits used are $\mathbf{b = 64, 128, 256, 512}$ respectively, from left to right. Curves for fixed time experiment are plotted in dashed lines. The fixed time plots are the same as the fixed bit plots for the proposed methods. For the baseline methods, the number of bits used in fixed time experiments are $\mathbf{b = 192, 384, 768, 1536}$ respectively, so that their running time is comparable with that of the proposed methods.

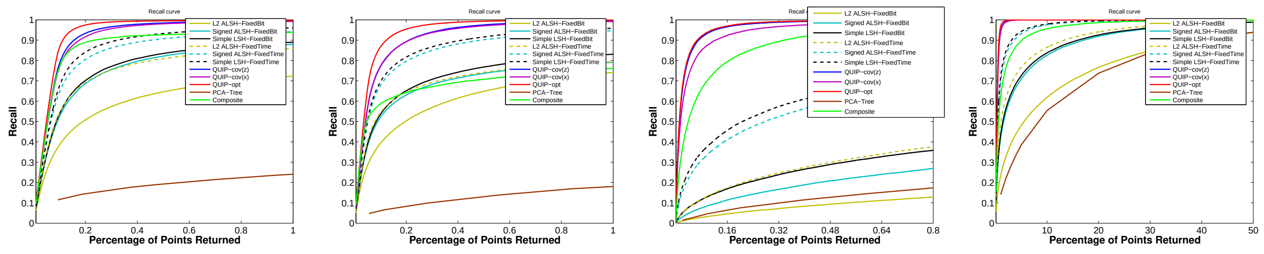


(a) ImageNet dataset, retrieval of Top 5 items.



(b) VideoRec dataset, retrieval of Top 50 items.

Figure 3: Precision Recall curves on ImageNet and VideoRec. See Figure 2 for more explanation.



(a) Movielens, top-10 (b) Netflix, top-10 (c) VideoRec, top-50 (d) ImageNet, top-5

Figure 4: Recall curves for different techniques (including *PCA-Tree* and *Composite Quantization* [10]) under different numbers of returned neighbors (shown as the percentage of total number of points in the database). We plot the recall curve instead of the precision recall curve because *PCA-Tree* uses original vectors to compute distances therefore the precision will be the same as recall in Top-K search. The number of bits used for all the plots is 512, except for *Signed ALSH-FixedTime*, *L2 ALSH-FixedTime* and *Simple LSH-FixedTime*, which use 1536 bits.

formance is not that large. In theory, the non-centered covariance matrix of the *held-out* samples (Σ_Z) can be quite different than that of the database (Σ_X), leading to drastically different results. However, the comparable performance implies that it is often safe to use Σ_X when learning a codebook. On the other hand, when a small set of *held-out* samples is available, *QUIP-opt* outperforms both *QUIP-cov(x)* and *QUIP-cov(z)* on all four datasets. This is because it learns the codebook with constraints that steer learning towards retrieving the maximum dot product neighbors in addition to minimizing the quantization error. The overall training for *QUIP-opt* was quite fast, requiring 1 to 10 minutes using a single-threaded implementation, depending on the dataset size.

6 Conclusion

We have described a quantization based approach for fast approximate inner product search, which relies on robust learning of codebooks in multiple subspaces. One of the proposed variants leads to a very simple kmeans-like learning procedure and yet outperforms the existing state-of-the-art by a significant margin. We have also described its theoretical properties including the concentration bounds. We introduced novel constraints in the quantization error minimization framework that lead to even better codebooks, tuned to the problem of highest dot-product search. Extensive experiments on retrieval and classification tasks show the advantage of the proposed method over the existing techniques. In the future, we would like to analyze the theoretical guarantees associated with the constrained optimization procedure.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006.

[2] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 931–938. IEEE, 2014.

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

[4] J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[5] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems*, 1994.

[6] E. Cohen and D. D. Lewis. Approximating matrix multiplication for pattern recognition tasks. *Journal of Algorithms*, 30(2):211–252, 1999.

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

[8] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[9] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2013.

[10] C. Du and J. Wang. Inner product similarity search using compositional codes. *CoRR*, abs/1406.4966, 2014.

[11] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, Jan. 2011.

[12] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2329–2336. IEEE, 2014.

[13] N. Koenigstein, P. Ram, and Y. Shavitt. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 535–544, New York, NY, USA, 2012. ACM.

[14] K. Neammanee and P. Thongtha. Improvement of the non-uniform version of berry-esseen inequality via paditz-siganov theorems. *Journal of Inequalities in Pure and Applied Mathematics (JIPM)*, 8(4), 2007.

[15] B. Neyshabur and N. Srebro. A simpler and better lsh for maximum inner product search (mips). *arXiv preprint arXiv:1410.5518*, 2014.

[16] P. Ram and A. G. Gray. Maximum inner-product search using cone trees. In *In SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*, 2012.

[17] M. Sabin and R. Gray. Product code vector quantizers for waveform and voice coding. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 32(3):474–488, Jun 1984.

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

[19] A. Shrivastava and P. Li. An improved scheme for asymmetric LSH. *CoRR*, abs/1410.5410, 2014.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *ArXiv e-prints*, Sept. 2014.

[21] T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Sparse composite quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4548–4556, 2015.