A Biresolution Spectral framework for Product Quantization

Lopamudra Mukherjee
University of Wisconsin-Whitewater
mukherjl@uww.edu

Sathya N. Ravi
University of Wisconsin-Madison
ravi5@wisc.edu

Jiming Peng
University of Houston
jopeng@uh.edu

Vikas Singh
University of Wisconsin-Madison
vsingh@biostat.wisc.edu

Abstract

Product quantization (PQ) (and its variants) has been effectively used to encode high-dimensional data into compact codes for many problems in vision. In principle, PQ decomposes the given data into a number of lower-dimensional subspaces where the quantization proceeds independently for each subspace. While the original PQ approach does not explicitly optimize for these subspaces, later proposals have argued that the performance tends to benefit significantly if such subspaces are chosen in an optimal manner. Despite such consensus, existing approaches in the literature diverge in terms of which specific properties of these subspaces are desirable and how one should proceed to solve/optimize them. Nonetheless, despite the empirical support, there is less clarity regarding the theoretical properties that underlie these experimental benefits for quantization problems in general. In this paper, we study the quantization problem in the setting where subspaces are orthogonal and show that this problem is intricately related to a specific type of spectral decomposition of the data. This insight not only opens the door to a rich body of work in spectral analysis, but also leads to distinct computational benefits. Our resultant biresolution spectral formulation captures both the subspace projection error as well as the quantization error within the same framework. After a reformulation, the core steps of our algorithm involve a simple eigen decomposition step, which can be solved efficiently. We show that our method performs very favorably against a number of state of the art methods on standard data sets.

1. Introduction

Nearest neighbor search is a fundamental problem in computer vision [32, 25]. With a rapid increase in sizes of the datasets/repositories that numerous end applications in vision leverage towards various tasks, mechanisms that offer accurate retrieval of nearest neighbors (approximately) from massive datasets are seeing renewed interest [41, 21] — both in terms of their theoretical/empirical properties as well as how such schemes can be translated to novel architectures (such as small form factor devices). At the high level, the goal is to retrieve (one or more) nearest neighbors $N(x) = \text{argmin}_y D(x, y)$ for a query image x, where $D(\cdot, \cdot)$ is a placeholder function for measuring distances (or similarities) between a pair of images. The classical strategy is to use hashing-based ideas from theoretical computer science, specifically, Euclidean Locality sensitive hashing (LSH) [9], which offers nice performance guarantees. However, there is a general consensus within the vision community that empirically, certain simple schemes that are informed by the distribution of the data, such as hierarchical k-means often work better [34, 40]. These considerations have led to the development of a parallel body of work on high dimensional indexing schemes in computer vision that are better tailored to the requirements posed by problems/applications we encounter in practice.

A widely used technique that emerged in computer vision, partly out of the issues described above, is *vector quantization* [33, 18]. The basic premise here is that high-dimensional vectorial representations of the image (derived from certain upstream feature extraction methods) can be encoded into a compact code and indexed. Then, when provided a query vector, we similarly quantize it into a codeword – the distances between codewords thereby serve as a surrogate for exhaustively calculating the original distances. The idea hits a sweet spot balancing retrieval performance and efficiency. For instance, mapping the distributions into a short code keeps the memory footprint small, enabling one to index/query from billion-sized dataset stored within the main memory. Second, if the codeword embedding is sensible, we obtain sizable gains in calculating distances, with a minor dependence on the ambient dimensionality d. When used together with inverted indexing [4], we can perform high-quality search on massive datasets in near real time.

The success of vector quantization algorithms initiated a rich and still evolving set of developments on how to derive codes that are best informed by the properties of the data/vector distribution in the native space. Due to these developments, there are a broad suite of algorithms on distance-preserving codes [9] that have been shown to work well in practice. At the high level, extensions of the quantization idea in recent years can be categorized under two broad classes. Several algorithms view quantization via the lens of binary hashing [9, 17, 42]; that is, given the set of high dimensional vectors, we seek to identify an embedding on the $\{0, 1\}$ hypercube such that the original distance (or similarity) between each pair of vectors is captured to high fidelity by the Hamming distance between the corresponding binary codes. Separately, in Product Quantization (PQ) methods [22], a data vector is vector-quantized to its nearest 'codeword' in a codebook (and one typically operates with a large number of codewords). Unlike hashing, PQ methods do not always quantize the query vector – instead, the distance between a pair of vectors is approximated by the distance between their codewords. One advantage of PQ methods is that they reduce quantization noise leading to improvements in search quality. Part of the reason is the additional degrees of freedom — the number of possible distances in quantization based coding is significantly higher compared to the Hamming distance setup used in binary hashing (which only permits discrete distances). Further, PQ is attractive for large-scale applications since it has a small computational footprint: pre-computed distances between codewords can be stored in tables, and a query merely involves table lookups using codeword indices.

Scope/rationale of this paper. Product Quantization can be conveniently thought of as a decomposition of the original vector space into the Cartesian product of a given number of low-dimensional subspaces [22]. PQ achieves this by partitioning the feature space into disjoint subsets of features and projecting the dataset into the subspaces spanned by these feature subsets. But the formulation and solution scheme offers little clarity on the optimality of such subspaces in terms of quantization error — the central objective of interest. Motivated by this observation, recently some papers have studied the problem of finding subspaces that explicitly seek to minimize the quantization error. Perhaps the most prominent result within these papers is the one called Cartesian Kmeans [35] and a similar idea developed independently [14]. Cartesian Kmeans expresses each region center as an additive combination of multiple subcenters. Each subcenter is chosen from a set of subcenters lying in a given subspace. These collection of subspaces are considered orthogonal. But more recently, [5] argued against the need for such strong orthogonality constraints and provided compelling evidence that their new proposal model (called Additive Quantization) offers improved performance relative to PQ without including orthogonal constraints, even when it uses the same additive model as in Cartesian Kmeans. Despite these benefits, Additive Quantization (AQ) remains computationally expensive to deploy on large scale datasets. The key reason is that the solving the underlying encoding optimization problem in AQ turns out to be equivalent to well-known combinatorially hard problems. Separately, the distance calculation workload in AQ increases significantly relative to PQ. The premise of our paper is to investigate whether the optimization problem in PQ can be endowed with additional structure to improve its performance *without incurring* the corresponding computational/efficiency bottleneck.

Our contributions. In this paper, we reformulate Product Quantization as the problem of finding multiple biresolution matrix factorizations of a correlation matrix of the given dataset, subject to orthogonality of the subspaces. Surprisingly, we find that the orthogonality property endows the problem with nice structure and leads to problems that are efficiently solvable. We show that under our reformulation, the core modules in the quantization problem reduce to **(i)** a spectral decomposition of a (set of) matrices followed by **(ii)** a set of orthogonal matrix factorizations, which can be done independently of each other. The **main contribution** of this paper is: a) to show that both subproblems are related (in distinct ways) to spectral analysis of the underlying data vectors leading to a bilevel formulation. To our knowledge, the connection of spectral analysis to product quantization has not been established before, though the model in [14] does make use of eigen allocation to find a lower bound of the objective. b) To solve our model, we propose a block-coordinate descent scheme, the core component of which is finding the eigen vectors (and values) of a matrix (dependent on the dimensionality of the original data, *not* the number of samples). This process can be solved efficiently even for large datasets using fast (exact) eigen solvers or very fast approximate schemes. c) We show that our method performs favorably against a number of state of the art methods on large datasets, proving the efficacy of our model. We describe our framework in Section 2.

Background and Related Work. Here we summarize some of the most popular works on product quantization. But first, we describe the model for the Vector Quantization(VQ) problem. Let $X \in \mathbb{R}^{d\times n}$ be a high dimensional matrix representing n examples in d dimensions, and x_i (ith sample of X) is in $\mathbb{R}^d$. Then the objective becomes

$$\min_j \sum_{x_i \in X} \|x_i - c_j\|^2 \equiv \min_y \sum_{x_i \in X} \|x_i - Cy_i\|^2 \quad (1)$$

Here each $c_j \in \mathbb{R}^d$ is considered a codeword in a codebook $C \in \mathbb{R}^{d\times m}$ (m denotes the number of subspaces throughout the paper) and the vector $y_i \in \mathbb{R}^m$ is a binary vector that has only one entry 1 pertaining to the codeword assigned to x_i. Product Quantization[22] can be formulated under this framework (1) with the additional desiderata that each codeword comes from the Cartesian product of a finite number of sub-codebooks, such that distortion is minimized:

$$\min_j \sum_{x_i \in X} \|x_i - c_j\|^2 \text{ s.t. } c_j \in C = C_1 \times \ldots \times C_m \quad (2)$$

Several other approaches also optimize the same basic objective (1) with additional constraints [35, 14]. We describe the Cartesian Kmeans (CKmeans) [35] model next, since it is closely related with our work. The CKmeans objective is

$$\min_y \sum_{x_i \in X} \|x_i - \sum_j C_j y_j\|^2 \text{ s.t. } C_j^T C_{k(\neq j)} = I \quad (3)$$

Here each $C_j \in R^{d \times h}$ is an unique subspace matrix, with h columns, each of which is a subcenter. There are m subspaces in all ($C_{j=1:m}$). The vector y_j is still the binary indicator vector with a single non-zero entry. Essentially, CKmeans expresses each data item as a sum of m center vectors, where each is chosen from a different subspace having h centers each. The subspaces are mutually orthogonal.
Other important approaches for PQ include Optimized Product Quantization (OPQ) [14], which alternatively finds a rotation matrix which when applied on the data minimizes distortion followed by searching for optimal codebooks. [48] proposed an alternative method to solve CKmeans. Additive Quantization [5], decomposes a vector as a sum of parts, each from a separate codebook, without the orthogonality constraints. Recently [29] used iterative local search to improve the performance of AQ. Another approach is Composite Quantization (CQ) [46], where a vector is approximated using the composition of several parts selected from several dictionaries. A sparse version of CQ has been proposed as well [47]. Other relevant works on this topic include iterative quantization [15], distance-encoded PQ[20], tree quantization[7], inner product search quantization[48, 38], inverted multi-index quantization[4, 44], bilayer product quantization [6], hashing-quantization hybrid method called polysemous codes [12] and others [28, 30, 8, 19, 45, 2, 23].

2. An initial model based on factorization

2.1. Preliminaries

The main idea behind our model is to reformulate the quantization objective as a problem of biresolution factorization of a matrix subject to orthogonality constraints. Next, we show how to solve this multilevel factorization efficiently.
First we rewrite the model in (3) for a given x as $\|x - \sum_j C_j y_j\|^2$. We construct a matrix $\mathbf{Q}$, by concatenating for all j, the column of C_j pertaining to $y_j = 1$. That is $\mathbf{Q}(:, j) = C_j y_j$. Therefore,

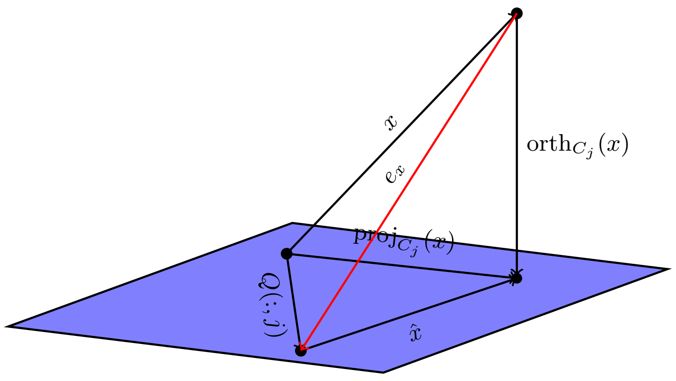

Figure 1. Relationship of x to subspace C_j

$$\min_{y_j} \|x - \sum_j C_j y_j\|^2 = \|x - \mathbf{Q}\mathbf{1}_m\|^2 \quad (4)$$
$$= x^T x - 2x^T \mathbf{Q}\mathbf{1}_m + \text{tr}(\mathbf{Q}^T\mathbf{Q})$$

The last term comes from the fact that $\mathbf{Q}^T\mathbf{Q}$ is a diagonal matrix (since its columns are from orthogonal subspaces). Here $\mathbf{1}_m$ is the vector of all ones of length m. Note that $x^T x$ is a constant, so minimizing $\|x - \sum_j \mathbf{C}_j y_j\|^2$ is the same as minimizing

$$\min_{\mathbf{Q}} \quad m x^T x - 2\sum_{j=1}^{m} x^T \mathbf{Q}(:, j) + \sum_{j=1}^{m} \mathbf{Q}(:j)^T \mathbf{Q}(:, j)$$
$$= \sum_{j=1}^{m} \|x - \mathbf{Q}(:, j)\|^2 = \sum_{j=1}^{m} \|x - C_j y_j\|^2 \quad (5)$$

This above manipulation shows that instead of computing the sum of the centers (one from each subspace), and then taking the squared norm distance to x as in (4), one can express the same objective as a sum of squares difference of x to each center vector in a given subspace (5). This decomposes the overall problem into an optimization over each of the subspaces independently.
Now, we analyze the objective (5) w.r.t. the terms that are relevant when finding the difference of a vector x to a particular subspace C_j. Note that $\mathbf{Q}(:, j)$ is the closest vector in C_j to x, among all possible centers. Let $\mathbf{e_x}$ denote the vector $x - \mathbf{Q}(:, j)$. $\to$ residual Let $\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x})$ be the projection of x onto C_j and $\text{orth}_{\boldsymbol{C_j}}(\boldsymbol{x})$ be the vector orthogonal to $\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x})$ as shown in Figure 1. Also let $\hat{\mathbf{x}} = \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x}) - \mathbf{Q}(:, j)$. Therefore, if $\mathbf{e}_x$ is the error vector (denoting the difference of the original vector x to its closest center), we obtain the following objective:

$$\|\mathbf{e}_x\|^2 = \|x - \mathbf{Q}(:, j)\|^2 = \text{orth}_{\boldsymbol{C_j}}(\boldsymbol{x})^2 + \hat{\mathbf{x}}^2 \quad (6)$$
$$= \text{orth}_{\boldsymbol{C_j}}(\boldsymbol{x})^2 + (\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x}) - \mathbf{Q}(:, j))^2$$

$\to$ figure

Taken over all x_i for a given subspace C_j (we extend the definition of $\mathbf{Q}$ to $\mathbf{Q}_i$ for each x_i), this reduces to

$$\sum_i \|\mathbf{e}_{x_i}\|^2 = \sum_i (\text{orth}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^2 + (\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i}) - \mathbf{Q}_i(:, j))^2) \quad (7)$$

Sum up

If the vector x_i and a subspace $\mathbf{C}_j$ is fixed, $\text{orth}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^2$ (and $\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})$) is also fixed. Therefore, the objective (7), in fact, seeking a vector $\mathbf{Q}_i(:,j)$), such that the second part of the objective is minimized. This is analogous to obtaining the K-means clustering over all $\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})$ on a given subspace C_j. Assume that c_k is a cluster in C_j where $k \in \{1 \cdots h\}$ (h is the number of clusters) to which the vector $\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})$ is assigned. The centroid can be found as a mean of all the vectors assigned to the cluster. Therefore,

$$\mathbf{Q}_i(:,j) = \sum_{\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i}) \in c_k} \frac{\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})}{n_k} \tag{8}$$

which is equivalent to using K-means for the VQ problem, where n_k is the number of points in the cluster. From this reasoning, we get

$$\sum_i (\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i}) - \mathbf{Q}_i(:,j))^2$$

$$= \underline{\sum_i \|\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})\|^2 - \sum_{k=1}^{h} \frac{1}{n_k} \sum_{i,l \in c_k} \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^T \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_l})} \tag{9}$$

Simply substituting this in (7),

$$\begin{aligned}\sum_i \|\mathbf{e}_{x_i}\|^2 &= \sum_i \text{orth}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^2 + \sum_i \|\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})\|^2 \\ &\quad - \sum_{k=1}^{h} \frac{1}{n_k} \sum_{i,l \in c_k} \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^T \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_l}) \\ &= \sum_i \|x_i\|^2 - \sum_{k=1}^{h} \frac{1}{n_k} \sum_{i,l \in c_k} \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^T \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_l}) \\ &= \text{constant} - \boxed{\sum_{k=1}^{h} \frac{1}{n_k} \sum_{i,l \in c_k} \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^T \text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_l})}\end{aligned}$$

By inspection, we see that minimizing the sum of norms of all $\mathbf{e}_{x_i}$ is equivalent to maximizing the pairwise dot-product of the projection of the vectors assigned to the same cluster.

Identifying Subspaces. So far, we have not optimized for the subspace itself. To do this, first let $\mathbf{V}_j$ be the orthonormal basis of C_j. Then, the projection operation can be expressed as a matrix multiplication with a matrix $\boxed{\mathbf{P}_j}$, where $\mathbf{P}_j = \mathbf{V}_j(\mathbf{V}_j^T\mathbf{V}_j)^{-1}\mathbf{V}_j^T = \mathbf{V}_j\mathbf{V}_j^T$. That is, $\boxed{\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x}) = \mathbf{P}_j x.}$ Therefore $\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_i})^T\text{proj}_{\boldsymbol{C_j}}(\boldsymbol{x_l}) = x_i^T\mathbf{P}_j^T\mathbf{P}_j x_l = x_i^T\mathbf{P}_j x_l$. The last identity is true because we know $\mathbf{P}^T\mathbf{P} = \mathbf{P}$ holds for a projection matrix $\mathbf{P}$. Therefore, the objective can be written as

$$\begin{aligned}&\max_{\mathbf{P}_j} \sum_{k=1}^{h} \frac{1}{n_k} \sum_{i,l \in c_k} x_i^T \mathbf{P}_j x_l \\ \equiv &\max_{\mathbf{V}_j} \sum_{k=1}^{h} \frac{1}{n_k} \sum_{i,l \in c_k} x_i^T \underline{\mathbf{V}_j\mathbf{V}_j^T} x_l \quad \text{s.t.} \quad \mathbf{V}_j^T\mathbf{V}_j = I\end{aligned}$$

We now introduce an indicator matrix $\mathbf{H}_j$ of size $n \times h$ to indicate the cluster membership. For each x_i in cluster c_k, $\mathbf{H}_j(i,k) = \frac{1}{\sqrt{n_k}}$ and 0 otherwise. Therefore, such a cluster indicator matrix is non-negative and orthogonal $\mathbf{H}_j^T\mathbf{H}_j = I, \mathbf{H}_j \geq 0$, an idea also independently used in [11]. Therefore, the previous model can be written as

$$\begin{aligned}&\max_{\mathbf{V}_j} \sum_{k=1}^{h} \frac{1}{n_k} \boxed{\sum_{i,l \in c_k} x_i^T \mathbf{V}_j \mathbf{V}_j^T x_l} \\ \equiv &\max_{\{\mathbf{V}_j,\mathbf{H}_j\}} \underline{\text{tr}(\mathbf{H}_j^T\mathbf{X}^T\mathbf{V}_j\mathbf{V}_j^T\mathbf{X}\mathbf{H}_j)} \\ &\text{s.t.} \quad \mathbf{V}_j^T\mathbf{V}_j = I, \quad \mathbf{H}_j^T\mathbf{H}_j = I, \quad \mathbf{H}_j \geq 0\end{aligned} \tag{10}$$

Let $\mathbf{X}$ be a $d \times n$ matrix (n points in d dimensions) with its ith column being equal to x_i. Essentially, $A_j = \mathbf{X}^T\mathbf{V}_j\mathbf{V}_j^T\mathbf{X}$, is the matrix of pairwise dot-product affinities of all data points after projection onto subspace C_j. The objective can also be rewritten as $\text{tr}(\mathbf{H}_j^T\mathbf{X}^T\mathbf{V}_j\mathbf{V}_j^T\mathbf{X}\mathbf{H}_j) = \text{tr}(\mathbf{V}_j^T\mathbf{X}(\mathbf{H}\mathbf{H}^T)\mathbf{X}^T\mathbf{V}_j)$. Therefore the final model is

$$\boxed{\begin{aligned}&\max_{\{\mathbf{V}_j,\mathbf{H}_j\}} \sum_{j=1}^{m} \text{tr}(\mathbf{H}_j^T\mathbf{X}^T\mathbf{V}_j\mathbf{V}_j^T\mathbf{X}\mathbf{H}_j) \\ &\text{s.t.} \quad \mathbf{V}_j^T\mathbf{V}_j = I, \quad \mathbf{H}_j^T\mathbf{H}_j = I, \quad \mathbf{H}_j \geq 0\end{aligned}} \tag{11}$$

Sum up

Loss for whole space

3. Asking for orthogonal subspaces

We now incorporate the relationship among the subspaces as a constraint in the above model. Similar to CKmeans, we ask that the subspaces be mutually orthogonal, written as

$$\begin{aligned}&\max_{\{\mathbf{V}_j,\mathbf{H}_j\}} \sum_{j=1}^{m} \text{tr}(\mathbf{H}_j^T\mathbf{X}^T\mathbf{V}_j\mathbf{V}_j^T\mathbf{X}\mathbf{H}_j) \\ &\text{s.t.} \quad \mathbf{V} = [\mathbf{V}_1 \ldots \mathbf{V}_m], \ \mathbf{V}^T\mathbf{V} = \mathbf{H}_j^T\mathbf{H}_j = I, \ \mathbf{H}_j \geq 0\end{aligned}$$

where $\mathbf{V}$ is created by concatenating $\mathbf{V}_1$ to $\mathbf{V}_m$. This makes the model for quantization analogous to *biresolution matrix factorization* with orthogonality constraints. Such ideas related to multi-resolution matrix factorizations have been studied recently [24] and shown to have nice theoretical properties. To solve this model in (12), we fix one set of variables at each time and solve for the other.

3.1. Optimizing Subspace Projection Error: $\mathbf{H}_j$ fixed

Here, we assume that the $\mathbf{H}_j$ is fixed (given to us from the previous iterations or initializations). When $\mathbf{H}_j$'s are fixed, we can express $\mathbf{C}_j = \mathbf{X}(\mathbf{H}_j\mathbf{H}_j^T)\mathbf{X}^T$ (note that boldface $\mathbf{C}_j$ is different from C_j notation used earlier, which pertains to the subspace matrix) which is constant and of size $d \times d$. Therefore, the model can be written as

$$\max_{\mathbf{V}} \sum_{j=1}^{m} \text{tr}(\mathbf{V}_j^T\mathbf{C}_j\mathbf{V}_j) \ \text{ s.t. } \ \mathbf{V}^T\mathbf{V} = I \tag{12}$$

Note that for $m = 1$, this is equivalent to finding the principle components of a given matrix. For $m > 1$, this is equivalent to a simultaneous PCA of m covariance matrices, with the additional requirement that each such principal component is mutually orthogonal. We first show that this model can be solved efficiently and optimally using a SDP formulation. To simplify presentation, we discuss the case where $m = 2$ followed by the more general case.

Two subspace ($m = 2$) case. WLOG, we consider $j = [1, 2]$. Our goal is to find an orthogonal matrix $\mathbf{V} = [v_1..v_d]$, assuming d is even such that the following is maximized:

$$\max_{\mathbf{V}_1,\mathbf{V}_2} \operatorname{tr}(\mathbf{V}_1^T\mathbf{C}_1\mathbf{V}_1) + \operatorname{tr}(\mathbf{V}_2^T\mathbf{C}_2\mathbf{V}_2) \quad (13)$$
$$\text{s.t. } \mathbf{V}^T\mathbf{V} = I, \quad \mathbf{V} = [\mathbf{V}_1 \quad \mathbf{V}_2]$$

We can divide the vectors of $\mathbf{V}$ into the set $\mathbf{V}_1 = [v_1 \dots v_{\frac{d}{2}}]$ and $\mathbf{V}_2 = [v_{\frac{d}{2}+1} \dots v_d]$. The model above can be equivalently written as follows:

$$\max_{\mathbf{W}_1,\mathbf{W}_2} \operatorname{tr}(\mathbf{C}_1\mathbf{W}_1) + \operatorname{tr}(\mathbf{C}_2\mathbf{W}_2) \quad (14)$$
$$\text{s.t. } \mathbf{W}_1 + \mathbf{W}_2 = I, \quad \mathbf{W}_1\mathbf{W}_2 = 0$$
$$\operatorname{rank}(\mathbf{W}_i) = \frac{d}{2}, \quad \mathbf{W}_i \succeq 0 \quad \forall i$$

The above transformation is possible if we set $\mathbf{W}_1 = \sum_{i=1}^{\frac{d}{2}} v_i v_i^T$ (equivalently $\mathbf{W}_2 = \sum_{i=\frac{d}{2}+1}^{d} v_i v_i^T$). Since the sum of outer products of a set of orthogonal vectors is identity ($\sum_{i=1,\dots,d} v_i v_i^T = I$), it follows that $\mathbf{W}_1 + \mathbf{W}_2 = I$. The rank constraint in the above model, is computationally challenging to solve, therefore we relax it with the trace equality. In addition, what makes the model difficult is the constraint $\mathbf{W}_1\mathbf{W}_2 = 0$. In the following, we show that this constraint can be dropped and the *model can still be solved to optimality*. To see this, we consider the following:

$$\max_{\mathbf{W}_1,\mathbf{W}_2} \operatorname{tr}(\mathbf{C}_1\mathbf{W}_1) + \operatorname{tr}(\mathbf{C}_2\mathbf{W}_2) \quad (15)$$
$$\text{s.t. } \mathbf{W}_1 + \mathbf{W}_2 = I, \quad \operatorname{tr}(\mathbf{W}_i) = \frac{d}{2}, \quad \mathbf{W}_i \succeq 0 \ \forall i$$

The above semidefinite programming problem has an interesting property, informally, the following theorem shows that the nonconvex constraint $\mathbf{W}_1\mathbf{W}_2 = 0$ can be ignored.

Theorem 3.1. *Let* $(\mathbf{W}_1^*, \mathbf{W}_2^*)$ *be the optimal solution to problem (15). Then it holds* $\mathbf{W}_1^*\mathbf{W}_2^* = 0$.

Proof: Since both $\mathbf{W}_1^*$ and $\mathbf{W}_2^* = I - \mathbf{W}_1^*$ are positive semi-definite. Let $\lambda_1, \lambda_2, \cdots, \lambda_d$ are the eigenvalues of $\mathbf{W}_1^*$ and $v_1, v_2, \cdots, v_d$ be the corresponding eigenvectors. It can be shown that the eigen values of $\mathbf{W}_2$ are exactly $(1 - \lambda_i)$ for all $i = 1 \dots d$ (we define $\hat{v}_i$ as the corresponding eigen vector of $\mathbf{W}_2$). It holds

$$\mathbf{W}_1^* = \sum_{i=1}^{d} \lambda_i v_i v_i^T, \ \mathbf{W}_2^* = \sum_{i=1}^{d} (1 - \lambda_i)\hat{v}_i\hat{v}_i^T \quad (16)$$
$$\lambda_i \in [0, 1] \ \forall i = 1, \cdots, n$$

Using the above notation, we can rewrite problem (11) as

$$\max_{\lambda_i} \sum_{i=1}^{n} \lambda_i (v_i^T \mathbf{C_1} v_i - \hat{v}_i^T \mathbf{C}_2 \hat{v}_i) + \sum_{i=1}^{n} \hat{v}_i^T \mathbf{C}_2 \hat{v}_i \quad (17)$$
$$\text{s.t. } \sum_{i=1}^{n} \lambda_i = \frac{d}{2}, \lambda_i \in [0, 1], \forall i = 1, \cdots, n$$

Without loss of generality, we assume that the sequence $\{v_i^T \mathbf{C_1} v_i - \hat{v}_i^T \mathbf{C}_2 \hat{v}_i\}$ is sorted based on the decreasing order. Then, we can see that the maximal value of the above problem can be achieved by setting $\lambda_i = 1, i = 1, \cdots, \frac{d}{2}$. This shows that the matrix $\mathbf{W}_1^*$ is a idempotent matrix whose eigenvalues have values 0 or 1 (and so is $\mathbf{W}_2^*$). This implies $\mathbf{W}_j^{*2} = \mathbf{W}_j^*$ for $j = [1, 2]$. Since $(\mathbf{W}_1^* + \mathbf{W}_2^*)^2 = I$, we immediately have $\mathbf{W}_1^*\mathbf{W}_2^* = 0$. Note that for idempotent matrices, its rank equals the trace of the matrix. Therefore, the model in (15), satisfies all the constraints of (14).

General case. We now extend the above result to the case with multiple ($m > 2$) subspaces. For simplicity of discussion, we assume $r = \frac{d}{m} = \lfloor \frac{d}{m} \rfloor$.

Lemma 3.2. *At optimality, we have that* $\operatorname{rank}(\mathbf{W}_i) = r$.

Proof. (Sketch) Let us consider the following problem:

$$\max_{\mathbf{W}_i} \sum_{i=1}^{m} \operatorname{tr}(\mathbf{C}_i\mathbf{W}_i) \quad (18)$$
$$\text{s.t.} \sum_{i=1}^{m} \mathbf{W}_i = I, \operatorname{tr}(\mathbf{W}_i) = r \quad \forall i = 1, \cdots, m-1$$
$$\mathbf{W}_i \succeq 0 \quad \forall i = 1, \cdots, m$$

Note that given the first two constraints, the trace constraint on $\mathbf{W}_m$ is satisfied automatically. The dual problem can be written as

$$\min_{\mathbf{Y}, z_i} \operatorname{tr}(\mathbf{Y}) + r \sum_{i=1}^{m-1} z_i \quad (19)$$
$$\text{s.t. } \mathbf{Y} + z_i I - \mathbf{S}_i = \mathbf{C}_i, \ \mathbf{S}_i \succeq 0, \ i = 1, \cdots, m-1$$
$$\mathbf{Y} - \mathbf{S}_m = \mathbf{C}_m, \quad \mathbf{S}_m \succeq 0$$

One can verify that both problems(18) and (19) are strictly feasible. Let $(\mathbf{W}_1^*, \cdots, \mathbf{W}_m^*)$ and $(\mathbf{Y}^*, \mathbf{S}_1^*, \cdots, \mathbf{S}_m^*, z_1^*, \cdots, z_{m-1}^*)$ be the optimal solution to the primal (18) and the dual (19) problem respectively. Using the duality theorem for semidefinite programming, we have

$$\mathbf{W}_i^*\mathbf{S}_i^* = \mathbf{S}_i^*\mathbf{W}_i^* = 0, \quad \forall i = 1, \cdots, m \quad (20)$$

We extend z with $z_m^* = 0$. It follows from the constraints of the dual problem that

$$\mathbf{W}_i^*(\mathbf{Y}^* + z_i^* I - \mathbf{C}_i) = 0, \quad \forall i = 1, \cdots, m \quad (21)$$

Denote $\mathbf{W}_i^* = \mathbf{V}_i \mathbf{\Lambda}_i \mathbf{V}_i^T$, where Λ_i is a diagonal matrix in a suitable space whose diagonal elements are the eigenvalues of $\mathbf{W}_i^*$, and $\mathbf{V}_i$ is a matrix whose columns are the eigenvectors corresponding to the eigenvalues of $\mathbf{W}_i^*$ satisfying $\mathbf{V}_i^T \mathbf{V}_i = I_{d_i}$ where d_i is the number of positive eigenvalues of $\mathbf{W}_i^*$. We therefore have

$$\mathbf{V}_i^T (\mathbf{Y}^* - \mathbf{C}_i) \mathbf{V}_i = -z_i^* I_{d_i}, \quad \forall i = 1, \cdots, m \tag{22}$$

The rest of the proof is constructive, which shows that there exists an algorithm which solves (22). In particular, the algorithm produces a sequence of dual variables $\mathbf{Y}$ and z from which a primal solution $\mathbf{W}_i$ can be extracted which always has rank r. We show that the sequence converges in the primal dual gap. Now, because we have strict feasibility and that the feasible set is compact, the limit of such a sequence also has rank r, so we get the desired result. Note that we do not require that $\mathbf{C}_i$ be full rank, only to be at least $r = \frac{d}{m}$, which can be satisfied based on the choice of m. We discuss this issue briefly in the supplement. □

Recall from the proof of lemma (3.2) that the update of $\mathbf{W}_i$, is done in a way such that the resultant matrix has rank r. This implies that the trace and rank of $\mathbf{W}_i$ are same. So, similar to the $m = 2$ case, we will show that $\mathbf{W}_i$'s are idempotent which implies that solving the convex problem (18) guarantees us a solution such that $\mathbf{W}_i \mathbf{W}_j = 0$.

Theorem 3.3. *Let $\mathbf{W}_i$, $i = 1 \ldots m$, be matrices satisfying $\sum_{i=1}^m \mathbf{W}_i = I$. Then $\mathbf{W}_i^2 = \mathbf{W}_i$, also implies $\mathbf{W}_i \mathbf{W}_j = 0, i \neq j$.*

3.2. Optimizing Quantization Error: $\mathbf{V}_j$ fixed

When the variable $\mathbf{V}_j$ are fixed, the problem reduces to m instances of the non-negative PCA problem.

$$I_{\mathbf{H}_j} = \max_{\mathbf{H}_j} \; \mathrm{tr}(\mathbf{H}_j^T \underbrace{\mathbf{X}^T \mathbf{V}_j \mathbf{V}_j^T \mathbf{X}}_{\mathbf{D}_j} \mathbf{H}_j) \tag{23}$$

$$\text{s.t.} \quad \mathbf{H}_j^T \mathbf{H}_j = I, \quad \mathbf{H}_j \geq 0$$

As before the matrix $\mathbf{D}_j = \mathbf{X}^T \mathbf{V}_j \mathbf{V}_j^T \mathbf{X}$ is given of size $n \times n$, but its rank is $\frac{d}{m}$. In addition to non-negativity, sparseness is also a desirable property for non-negative pca solutions. The general formulation of non-negative pca is known to be NP-complete. However, a recent paper by [3] show that if the rank of the matrix $\mathbf{D}_j$ is bounded (which is true in our case), then their algorithm, given a sparsity factor, can find the optimal non-negative principal component precisely. Though their algorithm is designed to recover only the top principal component, it can be extended to multiple principal components, by deflating $\mathbf{D}_j$ to remove the previous eigen vectors. For faster solutions to non-negative PCA problem, we can also use iterative multiplicative update schemes proposed in [10]. We adopt this approach to solve the above model.

4. Efficient Block Coordinate Descent Model

The SDP model in Section 3.1 is designed to solve the problem 12 optimally. However, standard SDP solvers do not scale well computationally to the large-scale datasets we consider here [27, 37]. Therefore, we study an alternative approach which is computationally much cheaper, but still has nice convergence guarantees. In particular, we design an efficient block coordinate descent algorithm using a simple observation of our SDP problem (18).
First we rewrite the model derived for the case $m = 2$ (Eq (15)). The optimization problem we solve is

$$\max_{\mathbf{W}_1, \mathbf{W}_2} \; \mathrm{tr}\left(\mathbf{C}_1 \mathbf{W}_1\right) + \mathrm{tr}\left(\mathbf{C}_2 \mathbf{W}_2\right) \tag{24}$$

$$\text{s.t. } \mathbf{W}_1 + \mathbf{W}_2 = I, \mathbf{W}_1, \mathbf{W}_2 \succeq 0, \mathrm{tr}\left(\mathbf{W}_i\right) = \frac{d}{2}$$

Eliminating $\mathbf{W}_2$ using the equality constraints, we get the equivalent problem as

$$\max_{\mathbf{W}_1} \mathrm{tr}\left(\mathbf{C}_1 \mathbf{W}_1\right) + \mathrm{tr}\left(\mathbf{C}_2 \left(I - \mathbf{W}_1\right)\right) \tag{25}$$

$$\text{s.t. } 0 \preceq \mathbf{W}_1 \preceq I, \mathrm{tr}\left(\mathbf{W}_1\right) = \frac{d}{2}$$

Denoting $\mathbf{C} := \mathbf{C}_1 - \mathbf{C}_2, \mathbf{W} := \mathbf{W}_1$, we get the following optimization problem

$$\max_{\mathbf{W}} \mathrm{tr}(\mathbf{C}\mathbf{W}) \quad \text{s.t. } 0 \preceq \mathbf{W} \preceq I, \mathrm{tr}(\mathbf{W}) = \frac{d}{2} \tag{26}$$

Observe that this exactly corresponds to computing the $\frac{d}{2}$ eigenvalues of $\mathbf{C}$ by Fan's theorem [31, 26] which can be efficiently computed.
Case $m > 2$: Now we will design a coordinate descent algorithm using the above observation to solve our optimization problem. Recall that setting $r = \frac{d}{m}$, the problem we seek to solve is

$$\max_{\mathbf{W}_i \succeq 0} \sum_{i=1}^{k} \mathrm{tr}\left(\mathbf{C}_i \mathbf{W}_i\right) \text{s.t. } \sum_i \mathbf{W}_i = I, \mathrm{tr}\left(\mathbf{W}_i\right) = r \tag{27}$$

Assume we have a feasible solution $\mathbf{W}_i$. We use two $\mathbf{W}_i$'s (say $\mathbf{W}_1, \mathbf{W}_2$) as our decision variables, keeping others fixed. Then, we solve the following:

$$\max_{\mathbf{W}_i \succeq 0} \sum_{i=1}^{2} \mathrm{tr}\left(\mathbf{C}_i \mathbf{W}_i\right) \text{s.t. } \sum_{i=1}^{2} \mathbf{W}_i = \mathbf{E}, \mathrm{tr}\left(\mathbf{W}_i\right) = r \tag{28}$$

where $\mathbf{E} = I - \sum_{i \neq 1,2} \mathbf{W}_i, 0 \preceq \mathbf{E} \preceq I$. We call this as subproblems $\mathcal{S}$. To solve $\mathcal{S}$, we formulate the equivalent eigen problem as follows.
Note that if we construct $\mathbf{E}$ for a feasible solution $\mathbf{W}$, $\mathbf{E}$ is a positive semidefinite matrix since $\mathbf{W}_i \preceq I \; \forall \; i$. Implicitly, subproblems (28) is a generalized eigenvalue problem so we can use a standard technique called the Fix-Heiberger deflation [36] to reduce it to a standard eigenvalue problem. First, we compute a decomposition of $\mathbf{E}$ as

$\mathbf{E} = \mathbf{U}^T\mathbf{U}$. This decomposition is always possible if we first do an eigenvalue decomposition of $\mathbf{E} = QDQ^T$ and set $\mathbf{U} = QD^{\frac{1}{2}}$. It follows that $\mathbf{U}\mathbf{E}\mathbf{U}^T = I_{2r}$. Since $\mathbf{E} = \mathbf{W}_1 + \mathbf{W}_2$, it follow that $\mathbf{U}(\mathbf{W}_1+\mathbf{W}_2)\mathbf{U}^T = I_{2r}$. Furthermore, $\mathbf{U}\mathbf{W}_1\mathbf{U}^T = I_{2r} - \mathbf{U}\mathbf{W}_2\mathbf{U}^T$. Therefore, $\mathbf{W}_1$ can be written as $\mathbf{W}_1 = \mathbf{U}^T(I_{2r} - \mathbf{U}\mathbf{W}_2\mathbf{U}^T)\mathbf{U} = \mathbf{U}^T(\mathbf{M})\mathbf{U}$. From the construction of $\mathbf{M}$, it is evident that $\mathbf{M} \preceq I$. Therefore, we can reformulate problem $\mathcal{S}$ in the following equivalent form which can be solved using an eigen decomposition:

$$\max_{\mathbf{M}} \operatorname{tr}(\mathbf{U}^T\mathbf{C}\mathbf{U}\mathbf{M}) \quad \text{s.t. } 0 \preceq \mathbf{M} \preceq I_{2r}, \operatorname{tr}(\mathbf{M}) = r$$

Now we write the block coordinate descent steps:

Algorithm 1 Block Coordinate Descent Method

```
Pick feasible points $\mathbf{W}_i$ or set $\mathbf{W}_i = \frac{d}{m}I$.
for $t = 0, 1, 2, \cdots, T$ do
    for $(i,j) \in \{1,2,\cdots,m\} \times \{1,2,\cdots,m\}$ do
        Fix all but $\mathbf{W}_i, \mathbf{W}_j$
        Solve subproblem $\mathcal{S}$ and update $\mathbf{W}_i, \mathbf{W}_j$.
    end for
end for
```

Lemma 4.1. (Convergence) *Algorithm 1 converges to the global optimal solution of problem (27).*

Proof. The subproblems are solved to an arbitrary accuracy $\epsilon \geq 0$ using eigendecomposition, hence the algorithm generates a monotonically decreasing sequence in the objective, see [16]. Now given that (27) is a convex problem with a compact feasible set, we have that every limit point of the sequence generated by Alg. 1 is a global minimizer, following the argument in theorem 3 of [43]. Note that Theorem 3 in [43] reinterprets the subproblems for the model with a log-det barrier function as solving a strictly convex probleme which can be extended to our problem as well. □

Complexity: $\mathcal{S}$ can be solved using an eigendecomposition. Since Alg. 1 will converge even when the subproblems are solved approximately, we can deploy state of the art techniques to solve $\mathcal{S}$. In particular, using the method in [13] computing the top eigenvector of $\mathbf{U}^T\mathbf{C}\mathbf{U}$ is $\tilde{\mathcal{O}}(\log 1/\epsilon)$ for an accuracy of ϵ. We can compute the top r eigenvalues approximately recursively, hence the complexity of solving $\mathcal{S}$ is $r\tilde{\mathcal{O}}(\log 1/\epsilon)$ and it takes $\mathcal{O}(n^3)$ flops to compute the LDL decomposition[1] of $\mathbf{E}$. So the complexity of algorithm 1 is $Tk^2\tilde{\mathcal{O}}\left(n^3 + r\log 1/\epsilon\right)$ compared to $T\tilde{\mathcal{O}}((nk)^4 \log 1/\epsilon)$ using interior point methods [39].

Comparison with Cartesian Kmeans (CKmeans)[35]: Our model offers new insights into the connections of orthogonal subspace based PQ to spectral decomposition of the data in various forms. [35] solves the same objective but via solving the orthogonal Procrustes subproblem, which is optimally solvable. Here, we highlight the main differences of our method with CKmeans. First, the convergence of our method is much faster than CKmeans. Our method converges in < 10 iterations in most case, whereas CKmeans generally takes many more iterations (≥ 30 in most cases). If CKmeans is run to convergence, the amount of time is generally a multiplicative factor(5 to 10) of what our algorithm takes on an average. The convergence comparison is shown in Figure 2 (top row, col 2). Furthermore, our model is more general since it can be extended to the following scenario. With some minor modifications, we can show that our model for optimizing V_j is robust to the presence of error in C_j's. This means that it is enough to get approximate solutions for problem (23) which is a desirable since solving problem (23) optimally is not easy. To our knowledge, the CKmeans model cannot be directly extended to such cases.

5. Experiments

We performed a number of experiments to evaluate the empirical performance of our algorithm. We compare our methods to 7 other approaches, including Product Quantization (PQ) [22], CKmeans [35], Orthogonal Kmeans (OK-means) [35], ITQ [15], Optimized Product Quantization [14] – both the parametric (OPQ-p) and non-parametric (OPQ-np) versions, Composite Quantization(CQ) [46] as well as Additive Quantization(AQ/APQ) [5]. Note that AQ is included only in a subset of the results, due to computational issues on larger datasets. We evaluate the algorithms on a number of datasets commonly used in vision, which vary in size and dimensionality. These include randomly generated datasets obeying Gaussian distributions as well as other well-known datasets such as Sift25K, Sift1M, Gist1M, Mnist, Cifar, VladLong[5] and Deep1M[5].

Parameters. For most settings in ANN experiments, we use either $m = \{4, 8, 16\}$ and $h = 256$. Specifically, Mnist and Deep1M results are with $m = 4$ and $m = 16$, while others (except Cifar ($m = 25$)) are with $m = 8$. The number of clusters (say, for $m = 8$) is $k = 256^8 = 2^{64}$, so a total of 64-bits are used in the encoding. This choice of parameters is fairly standard since it leads to small lookup tables, fast encoding and each subcenter index maps to one byte. For initialization, we use the subspaces using Eigen Allocation[14], which is a lower bound for quantization.

Design. Broadly, we evaluate two aspects: (i) how well do the obtained subspaces perform w.r.t. quantization error and (ii) how well do the set of obtained codes *approximate the nearest neighbors*. We describe these results next.

What is the quantization error? All compared methods (including ours) optimize the same general form of quantization distortion, therefore, we evaluate how they perform w.r.t. minimizing distortion. Distortion is computed as shown in (1). For our method, we project the data on to the subspaces obtained and then quantize the result using Kmeans. Distortion is calculated for other methods similarly. We rank each method (1 being the best), based on the

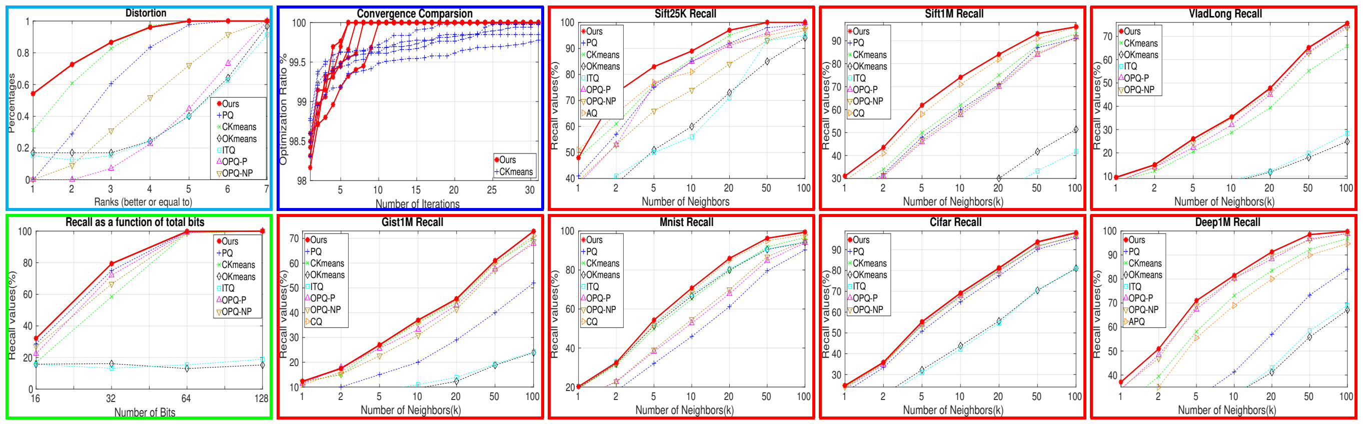

Figure 2. Top row: Left(cyan) shows the Ranking w.r.t. distortion. Col 2(blue) shows convergence wrt to ckmeans. NN Recall for Sift25K, SiftIM, VladLong are in Row 1 Col 3-5 (red) and for Gist1M, Mnist, Cifar and Deep1M in Row 2, Cols 2-5(red). Row 2, Col 1(green) shows the Recall w.r.t. bits.

distortion measure on the 40 instances of randomly generated data. Based on the results, we calculate the number of times each method obtains a certain rank or better. Figure 2 (row1, col1) shows the percentage of times each method obtains a particular (or higher) ranking. The results illustrate that our method obtains the rankings closer to the top more frequently than any other method, followed by CKmeans.

How well do we approximate nearest neighbors?

As a function of ANN search quality. To find ANNs, all algorithms perform linear scan search using asymmetric distance to compare a query with a database vector. The goodness of the approximate nearest neighbor search is measured in terms of Recall. For each query, we retrieve its k nearest neighbors and compute what fraction of ground-truth nearest neighbors are also found in the retrieved list of neighbors. The average recall score over all the queries is used as the measure for comparison. The ground-truth nearest neighbors are computed over the original features. Here, we report the performance with $k \in \{1, 2, 5, 10, 20, 50, 100\}$. Figure 2 shows the recall results on 7 datasets relative to all other methods. Note that the dimensionality of these datasets vary from 128 for the Sift data to 960 for Gist1M. We see that in all cases, our method is among the best performing algorithms. On the Sift datasets, it has the strongest recall improvement - about 3.5% and 2.5% improvement (for Sift25K and Sift1M respectively), averaged over all k, compared to the next best algorithm. For other datasets, average improvement over the next best algorithm is in the range of $0.5 - 2\%$. Furthermore, we tested the statistical significance of these improvements, and found that in 6 out of 7 cases, the improvements (based on a Kolmogorov Smirnov test) were statistically significant at the 5% significance level. Additionally, we observe the performance of our overall method is not affected by this dependence and is mostly stable w.r.t. to the dimensionality of the datasets. This is not true for (some of the) other methods, particularly PQ, whose performance is at least partly affected as the dimensionality grows.

As a function of the number of bits. We analyze our method's performance by varying the number of bits used. Here, we generate random datasets (32 dimensions) and quantize it with bits choosen from $\{16, 32, 64, 128\}$. Figure 2 (bottom row, column 1) shows the recall as a function of code length. As is expected, increasing the number of bits leads to better performance in NN search for all methods. But we should note that the improvement of our approach over the other algorithms is significant even when the code length is 32, whereas other methods need longer codes to achieve high recall. This shows that the advantages of our approach with moderate length code is more pronounced.

Running Time and Convergence. Figure 2 (row1, col2) shows convergence of our algorithm is quick (usually ≤ 10 iterations). The only computationally intensive step is solving for eigen values of matrices (size $d \times d$) multiple times ($T \times \binom{m}{2}$), where T is the number of iterations). Using fast eigen solvers, the eigen value decomposition, for $d = 1000$, only takes a few milliseconds (large scale approximate eigenvalue solvers are also available). Overall the training phase takes $2-4$ seconds (for training size 10^4).

6. Summary

We formulated efficient algorithms for the Product Quantization problem subject to orthogonality of subspaces. We showed that the PQ problem with these constraints reduces to a biresolution spectral framework, where the core steps involve finding the eigen bases of the underlying data, subject to certain transformations. Such a connection of PQ to spectral analysis is interesting and to our knowledge, unavailable in the literature. Besides the theoretical contribution, we show that this formulation leads to an efficient solution for the problem, which compares very well with the state of the art methods on a variety of benchmark datasets.

Acknowledgements This work is supported by NIH R01 AG040396 (VS), NSF CCF 1320755 (VS), NSF CGV 1219016 (LM), UW CPCP (U54 AI117924), NSF CMMI-1131690(JP), CMMI-1537712(JP) and CNS-1637347(JP).

References

[1] LDL factorization of symmetric matrices. http://www.physics.arizona.edu/~restrepo/475A/Notes/sourcea-/node66.html. 7

[2] F. André, A.-M. Kermarrec, and N. Le Scouarnec. Cache locality is not enough: high-performance nearest neighbor search with product quantization fast scan. *Proceedings of the VLDB Endowment*, 9(4):288–299, 2015. 3

[3] M. Asteris, D. S. Papailiopoulos, and A. G. Dimakis. Nonnegative sparse pca with provable guarantees. In *ICML*, 2014. 6

[4] A. Babenko and V. Lempitsky. The inverted multi-index. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3069–3076. IEEE, 2012. 1, 3

[5] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 2, 3, 7

[6] A. Babenko and V. Lempitsky. Improving bilayer product quantization for billion-scale approximate nearest neighbors in high dimensions. *arXiv preprint arXiv:1404.1831*, 2014. 3

[7] A. Babenko and V. Lempitsky. Tree quantization for large-scale similarity search and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4240–4248, 2015. 3

[8] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016. 3

[9] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004. 1, 2

[10] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135. ACM, 2006. 6

[11] C. H. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*. SIAM, 2005. 4

[12] M. Douze, H. Jégou, and F. Perronnin. Polysemous codes. In *European Conference on Computer Vision*, pages 785–801. Springer, 2016. 3

[13] D. Garber, E. Hazan, C. Jin, S. M. Kakade, C. Musco, P. Netrapalli, and A. Sidford. Faster eigenvector computation via shift-and-invert preconditioning. *CoRR*, 2016. 7

[14] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2953, 2013. 2, 3, 7

[15] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011. 3, 7

[16] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007. 7

[17] K. Grauman and R. Fergus. Learning binary hash codes for large-scale image search. In *Machine Learning for Computer Vision*, pages 49–87. Springer, 2013. 2

[18] R. Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984. 1

[19] Q.-Z. Guo, Z. Zeng, S. Zhang, G. Zhang, and Y. Zhang. Adaptive bit allocation product quantization. *Neurocomputing*, 171:866–877, 2016. 3

[20] J.-P. Heo, Z. Lin, and S.-E. Yoon. Distance encoded product quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2131–2138, 2014. 3

[21] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008. 1

[22] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, 2011. 2, 3, 7

[23] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2321–2328, 2014. 3

[24] R. Kondor, N. Teneva, and V. Garg. Multiresolution matrix factorization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1620–1628, 2014. 4

[25] N. Kumar, L. Zhang, and S. Nayar. What is a good nearest neighbors algorithm for finding similar patches in images? In *European conference on computer vision*, pages 364–378. Springer, 2008. 1

[26] M. Laurent and F. Vallentin. Semidefinite optimization. 2012. 6

[27] J. D. Lee, B. Recht, N. Srebro, J. Tropp, and R. R. Salakhutdinov. Practical large-scale optimization for max-norm regularization. In *Advances in Neural Information Processing Systems*, pages 1297–1305, 2010. 6

[28] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang. Structure sensitive hashing with adaptive product quantization. *IEEE transactions on cybernetics*, 46(10):2252–2264, 2016. 3

[29] J. Martinez, J. Clement, H. H. Hoos, and J. J. Little. Revisiting additive quantization. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016. 3

[30] Y. Matsui, T. Yamasaki, and K. Aizawa. Pqtable: Fast exact asymmetric distance neighbor search for product quantization using hash tables. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1940–1948, 2015. 3

[31] P. Moscato, M. G. Norman, and G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Math. Oper. Res.*, 23(2):339–358, Feb. 1998. 6

[32] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014. 1

[33] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Transactions on communications*, 36(8):957–971, 1988. 1

[34] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. IEEE, 2006. 1

[35] M. Norouzi and D. Fleet. Cartesian k-means. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3017–3024, 2013. 2, 3, 7

[36] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998. 6

[37] G. Pataki. Bad semidefinite programs: they all look the same. 6

[38] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. Tao Shen. Learning binary codes for maximum inner product search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4148–4156, 2015. 3

[39] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996. 7

[40] J. Wang, J. Wang, Q. Ke, G. Zeng, and S. Li. Fast approximate k-means via cluster closures. In *Multimedia Data Mining and Analytics*, pages 373–395. Springer, 2015. 1

[41] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2005. 1

[42] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009. 2

[43] Z. Wen, D. Goldfarb, and K. Scheinberg. Block coordinate descent methods for semidefinite programming. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 533–564. Springer, 2012. 7

[44] Y. Xia, K. He, F. Wen, and J. Sun. Joint inverted indexing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3416–3423, 2013. 3

[45] J. Yuan and X. Liu. Product tree quantization for approximate nearest neighbor search. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2035–2039. IEEE, 2015. 3

[46] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, pages 838–846, 2014. 3, 7

[47] 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. 3

[48] X. Zhang, F. X. Yu, R. Guo, S. Kumar, S. Wang, and S.-F. Chang. Fast orthogonal projection based on kronecker product. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2929–2937, 2015. 3