Optimized Product Quantization for Approximate Nearest Neighbor Search

Tiezheng Ge[1*] Kaiming He[2] Qifa Ke[3] Jian Sun[2]
[1]University of Science and Technology of China [2]Microsoft Research Asia [3]Microsoft Research Silicon Valley

Abstract

Product quantization is an effective vector quantization approach to compactly encode high-dimensional vectors for fast approximate nearest neighbor (ANN) search. The essence of product quantization is to decompose the original high-dimensional space into the Cartesian product of a finite number of low-dimensional subspaces that are then quantized separately. Optimal space decomposition is important for the performance of ANN search, but still remains unaddressed. In this paper, we optimize product quantization by minimizing quantization distortions w.r.t. the space decomposition and the quantization codebooks. We present two novel methods for optimization: a non-parametric method that alternatively solves two smaller sub-problems, and a parametric method that is guaranteed to achieve the optimal solution if the input data follows some Gaussian distribution. We show by experiments that our optimized approach substantially improves the accuracy of product quantization for ANN search.

1. Introduction

Approximate nearest neighbor (ANN) search is of great importance for many computer vision problems, such as retrieval [17], classification [2], and recognition [18]. Recent years have witnessed the increasing interest (*e.g*., [18, 20, 3, 10, 6]) in encoding high dimensional data into distance-preserving compact codes. With merely tens of bits per data item, compact encoding not only saves the cost of data storage and transmission, but more importantly, it enables efficient nearest neighbor search on large-scale datasets, taking only a fraction of a second for each nearest neighbor query [18, 10]

Hashing [1, 18, 20, 19, 6, 8] has been a popular approach to compact encoding, where the similarity between two data points is approximated by the Hamming distance of their hashed codes. Recently, product quantization (PQ) [10] was applied to compact encoding, where a data point is vector-quantized to its nearest codeword in a predefined codebook, and the distance between two data points is approximated by the distance between their codewords. PQ achieves a large effective codebook size with the Cartesian product of a set of small sub-codebooks. It has been shown to be more accurate than various hashing-based methods (*c.f*. [10, 3]), largely due to its lower quantization distortions and more precise distance computation using a set of small lookup tables. Moreover, PQ is computationally efficient and thus attractive for large-scale applications—the Cartesian product enables pre-computed distances between codewords to be stored in tables with feasible sizes, and query is merely done by table lookups using codeword indices. It takes about 20 milliseconds to query against one million data points for the nearest neighbor by *exhaustive* search.

To keep the size of the distance lookup table feasible, PQ decomposes the original vector space into the Cartesian product of a finite number of low-dimensional subspaces. It has been noticed [10] that the prior knowledge about the structures of the input data is of particular importance, and the accuracy of ANN search would become substantially worse if ignoring such knowledge. The method in [11] optimizes a Householder transform under an intuition that the data components should have balanced variances. It is also observed that a random rotation achieves similar performance [11]. But the optimality in terms of quantization error is unclear. Thus, optimal space decomposition for PQ remains largely an unaddressed problem.

In this paper, we formulate product quantization as an optimization problem that minimizes the quantization distortions by searching for optimal codebooks and space decomposition. Such an optimization problem is challenging due to large number of free parameters. We proposed two solutions. In the first solution, we split the problem into two sub-problems, each having a simple solver. The space decomposition and the codebooks are then alternatively optimized, by solving for the space decomposition while fixing the codewords, and vice versa. Such a solution is non-parametric in that it does not assume any priori information about the data distribution. Our second solution is a parametric one in that it assumes the data follows Gaussian distribution. Under such assumption, we show that the lower bound of the quantization distortion has an analytical

*This work is done when Tiezheng Ge is an intern at Microsoft Research Asia.

formulation, which can be effectively optimized by a simple ***Eigenvalue Allocation*** method. Experiments show that our two solutions outperform the original PQ [10] and other alternatives like transform coding [3] and iterative quantization [6], even when the prior knowledge about the structure of the input data is used by PQ [10].

Concurrent with our work, a very similar idea is independently developed by Norouzi and Fleet [14].

2. Quantization Distortion

In this section, we show that a variety of distance approximation methods, including k-means [13], product quantization [10], and orthogonal hashing [19, 6], can be formulated within the framework of *vector quantization* [7] where quantization distortion is used as the objective function. Quantization distortion is tightly related to the empirical ANN performance, and thus can be used to measure the "optimality" of a quantization algorithm for ANN search.

2.1. Vector Quantization

Vector quantization (VQ) [7] maps a vector $\mathbf{x} \in \mathbb{R}^D$ to a *codeword* $\mathbf{c}$ in a *codebook* $\mathcal{C} = \{\mathbf{c}(i)\}$ with i in a finite index set. The mapping, termed as a *quantizer*, is denoted by: $\mathbf{x} \rightarrow \mathbf{c}(i(\mathbf{x}))$. In information theory, the function $i(\cdot)$ is called an *encoder*, and function $\mathbf{c}(\cdot)$ is called a *decoder* [7].

The quantization *distortion* E is defined as:

$$E = \frac{1}{n} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \tag{1}$$

where $\|\cdot\|$ denotes the l_2-norm, n is the total number of data samples, and the summation is over all the points in the given sample set. Given a codebook $\mathcal{C}$, a quantizer that minimizes the distortion E must satisfy the first Lloyd's condition [7]: the encoder $i(\mathbf{x})$ should map any $\mathbf{x}$ to its nearest codeword in the codebook $\mathcal{C}$. The distance between two vectors can be approximated by the distances between their codewords, which can be precomputed offline.

2.2. Codebook Generation

We show that a variety of methods minimize the distortion w.r.t. to the codebook using different constraints.

K-means

If there is no constraint on the codebook, minimizing the distortion in Eqn.(1) leads to the classical k-means clustering algorithm [13]. With the encoder $i(\cdot)$ fixed, the codeword $\mathbf{c}$ of a given $\mathbf{x}$ is the center of the cluster that $\mathbf{x}$ belongs to—this is the second Lloyd's condition [7].

Product Quantization [10]

If any codeword $\mathbf{c}$ must be taken from the Cartesian product of a finite number of sub-codebooks, minimizing the distortion in Eqn.(1) leads to the product quantization method [10].

Formally, denote any $\mathbf{x} \in \mathbb{R}^D$ as the concatenation of M subvectors: $\mathbf{x} = [\mathbf{x}^1, ...\mathbf{x}^m, ...\mathbf{x}^M]$. For simplicity it is assumed [10] that the subvectors have common number of dimensions D/M. The Cartesian product $\mathcal{C} = \mathcal{C}^1 \times ... \times \mathcal{C}^M$ is the set in which a codeword $\mathbf{c} \in \mathcal{C}$ is formed by concatenating the M sub-codewords: $\mathbf{c} = [\mathbf{c}^1, ...\mathbf{c}^m, ...\mathbf{c}^M]$, with each $\mathbf{c}^m \in \mathcal{C}^m$. We point out that the objective function for PQ, though not explicitly defined in [10], is essentially:

$$\min_{\mathcal{C}^1,...,\mathcal{C}^M} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \tag{2}$$
$$s.t. \quad \mathbf{c} \in \mathcal{C} = \mathcal{C}^1 \times ... \times \mathcal{C}^M.$$

It is easy to show that $\mathbf{x}$'s nearest codeword $\mathbf{c}$ in $\mathcal{C}$ is the concatenation of the M nearest sub-codewords $\mathbf{c} = [\mathbf{c}^1, ...\mathbf{c}^m, ...\mathbf{c}^M]$ where $\mathbf{c}^m$ is the nearest sub-codeword of the subvector $\mathbf{x}^m$. So Eqn. (2) can be split into M separate subproblems, each of which can be solved by k-means in its corresponding subspace. This is the PQ algorithm.

The benefit of PQ is that it can easily generate a codebook $\mathcal{C}$ with a large number of codewords. If each sub-codebook has k sub-codewords, then their Cartesian product $\mathcal{C}$ has k^M codewords. This is not possible for classical k-means when k^M is large. PQ also enables fast distance computation: the distances between any two sub-codewords in a subspace are precomputed and stored in a k-by-k lookup table, and the distance between two codewords in $\mathcal{C}$ is simply the sum of the distances compute from the M subspaces.

Iterative Quantization [6]

If any codeword $\mathbf{c}$ must be taken from "the vertexes of a rotating hyper-cube," minimizing the distortion leads to a hashing method called Iterative Quantization (ITQ) [6].

The D-dimensional vectors in $\{-a, a\}^D$ are the vertices of an axis-aligned D-dimensional hyper-cube. Suppose the data has been zero-centered. The objective function in ITQ [6] is essentially:

$$\min_{R,a} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \tag{3}$$
$$s.t. \quad \mathbf{c} \in \mathcal{C} = \{\mathbf{c} \mid R\mathbf{c} \in \{-a, a\}^D\}, \quad R^{\mathrm{T}}R = I,$$

where R is an orthogonal matrix and I is an identity matrix.

The benefit of using a rotating hyper-cube as the codebook is that the squared Euclidean distance between any two codewords is equivalent to the Hamming distance between their indices. So ITQ is in the category of binary hashing methods [1, 20, 19]. Eqn.(3) also indicates that any orthogonal hashing method is equivalent to a vector quantizer. The length a in (3) does not impact the resulting hashing functions as noticed in [6], but it matters when we compare the distortion with other quantization methods.

2.3. Distortion as the Objective Function

The above methods all optimize the same form of quantization distortion, but subject to different constraints. This

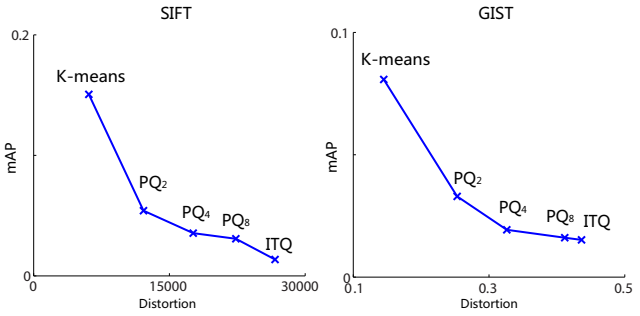

Figure 1: mAP *vs*. quantization distortion. We show results from five methods: k-means, ITQ, and three variants of PQ. The datasets are SIFT1M and GIST1M from [10]. All methods are given 16 bits for codeword length. The data consist of the largest 16 principal components (this is to enable measuring the ITQ distortion).

implies that distortion is an objective function that can be evaluated across different quantization methods. We empirically observe that the distortion is tightly correlated to the ANN search accuracy of different methods.

To show this, we investigate nearest neighbor search for 100 nearest neighbors on two large datasets. We adopt the common strategy of linear search using compact codes [10, 6]: the return results are ordered by their approximate distances to the query. Here both the query and the data are quantized, and their distance is approximated by their codeword distances (for ITQ this is equivalent to ranking by Hamming distance). We compute the mean Average Precision (mAP) over ten thousand and one thousand queries on the first and second data set, respectively. The mAP using different methods and their distortions are shown in Fig. 1. We compare five methods: k-means, ITQ, and three variants of PQ (decomposed into $M = 2$, 4, or 8 subspaces, denoted as PQ$_M$). For all methods the codeword length is fixed to be $B = 16$ bits, which essentially gives $K = 2^{16}$ codewords (though PQ and ITQ need not explicitly store them). We can see that mAP (from different methods) has a strong correlation with the quantization distortion. We observe similar behaviors under various ANN metrics, like precision/recall at the first N samples, with various number of ground-truth nearest neighbors.

Based on the above observations, we use distortion as an objective function to evaluate the optimality of a quantization method.

3. Optimized Product Quantization

Product quantization involves decomposing the D-dimensional vector space into M subspaces, and computing a sub-codebook for each subspace. M is determined by the budget constraint of memory space (to ensure a feasible lookup table size) and computational costs, and is predetermined in practice. We use an orthonormal matrix R to represent the space decomposition. The D-dimensional vector space is first transformed by R. The dimensions of the transformed space are then divided into D/M chunks. The i-th chunk, consisting of the dimensions of $(i-1) * D/M + \{1, 2, ..., D/M\}$, is then assigned to the i-th subspace. Note that any reordering of the dimensions can be represented by an orthonormal matrix. Thus R decides the dimensions of the transformed space assigned to each subspace. The free parameters of product quantization thus consist of the sub-codebooks for the subspaces, and the matrix R. The additional free parameters of R allows the vector space to rotate, thus relax the constraints on the codewords. From Fig. 1 we see that relaxing constraints leads to lower distortions. We optimize product quantization over these free parameters:

$$\min_{R,\mathcal{C}^1,...,\mathcal{C}^M} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{c}(i(\mathbf{x}))\|^2, \tag{4}$$
$$s.t. \quad \mathbf{c} \in \mathcal{C} = \{\mathbf{c} \mid R\mathbf{c} \in \mathcal{C}^1 \times ... \times \mathcal{C}^M, \;\; R^{\mathrm{T}}R = I\}$$

We call the above problem Optimized Product Quantization (OPQ). The effective codebook $\mathcal{C}$ is jointly determined by R and sub-codebooks $\{\mathcal{C}^m\}_{m=1}^M$.

Notice that assigning $\mathbf{x}$ to its nearest codeword $\mathbf{c}$ is equivalent to assigning $R\mathbf{x}$ to the nearest $R\mathbf{c}$. To apply the codebook in Eqn.(4) for encoding, we only need to preprocess the data by $R\mathbf{x}$, and the remaining step is the same as that in PQ.

Optimizing the problem in Eqn.(4) was considered "*not tractable*" [11], because of the large number of degrees of freedom. Previous methods pre-processed the data using simple heuristics like randomly ordering the dimensions [10] or randomly rotating the space [11]. The matrix R has not been considered in any optimization. In the following we propose a non-parametric iterative solution and a simple parametric solution to the problem of Eqn.(4).

3.1. A Non-Parametric Solution

Our non-parametric solution does not assume any data distribution [1]. We split the problem in Eqn.(4) into two simpler sub-problems that are optimized in an alternative way.

Step (i): Fix R, optimize the sub-codebooks $\{\mathcal{C}^m\}_{m=1}^M$.

Recall that assigning $\mathbf{x}$ to the nearest $\mathbf{c}$ is equivalent to assigning $R\mathbf{x}$ to the nearest $R\mathbf{c}$. Denote $\hat{\mathbf{x}} = R\mathbf{x}$ and $\hat{\mathbf{c}} = R\mathbf{c}$. Since R is orthonormal, we have $\|\mathbf{x}-\mathbf{c}\|^2 = \|\hat{\mathbf{x}}-\hat{\mathbf{c}}\|^2$. With R fixed, Eqn.(4) then becomes:

$$\min_{\mathcal{C}^1,...,\mathcal{C}^m} \sum_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}} - \hat{\mathbf{c}}(i(\hat{\mathbf{x}}))\|^2, \tag{5}$$
$$s.t. \quad \hat{\mathbf{c}} \in \mathcal{C}^1 \times ... \times \mathcal{C}^M.$$

[1] We follow the terminology in statistics that a "non-parametric" model is the one that does not rely on any assumption about the data distribution, while "parametric" model explicitly assumes certain parameterized distribution such as Gaussian distribution.

Eqn.(5) is the same problem as PQ in Eqn.(2). We can separately run k-means in each subspace to compute the sub-codebooks.

Step (ii): Fix the sub-codebooks $\{\mathcal{C}^m\}_{m=1}^M$, optimize R.
Since $\|\mathbf{x}-\mathbf{c}\|^2 = \|R\mathbf{x}-\hat{\mathbf{c}}\|^2$, the sub-problem becomes:

$$\min_R \sum_{\mathbf{x}} \|R\mathbf{x} - \hat{\mathbf{c}}(i(\hat{\mathbf{x}}))\|^2, \quad s.t. \quad R^{\mathrm{T}}R = I. \tag{6}$$

For each $\hat{\mathbf{x}}$, the codeword $\hat{\mathbf{c}}(i(\hat{\mathbf{x}}))$ is fixed in the subproblem and can be derived from the sub-codebooks computed in Step (i). To find $\hat{\mathbf{c}}(i(\hat{\mathbf{x}}))$, we simply concatenate the M sub-codewords of the sub-vectors in $\hat{\mathbf{x}}$. We denote $\hat{\mathbf{c}}(i(\hat{\mathbf{x}}))$ as $\mathbf{y}$. Given n training samples, we denote X and Y as two D-by-n matrices whose columns are the samples $\mathbf{x}$ and $\mathbf{y}$ respectively. Note Y is fixed in this subproblem. Then we can rewrite (6) as:

$$\min_R \|RX - Y\|_{\mathrm{F}}^2, \quad s.t. \quad R^{\mathrm{T}}R = I, \tag{7}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. This is the Orthogonal Procrustes problem [16, 6] and there is a closed-form solution: first apply Singular Value Decomposition to $XY^{\mathrm{T}} = USV^{\mathrm{T}}$, and then let $R = VU^{\mathrm{T}}$. In [6] this solution was used to optimized the ITQ problem in Eqn.(3).

Our algorithm alteratively optimizes Step (i) and (ii). Note that in Step (i) we need to run k-means, which by itself is an iterative algorithm. However, we notice that after Step (ii) the updated matrix R would not change the cluster membership of the previous k-means clustering results, thus we only need to refine the previous k-means results instead of restarting k-means. With this strategy, we empirically find that even if we only run one k-means iteration in each Step (i), our entire algorithm still converges to a good solution. A pseudo-code is in Algorithm 1.

Note that if we ignore line 3 and line 8 in Algorithm 1, it is equivalent to PQ (for PQ one might usually put line 2 in the inner loop). Thus its complexity is comparable to PQ, except that in each iteration our algorithm updates R and transforms the data by R. Fig. 2 shows the convergence of our algorithm. In practice we find 100 iterations are good enough for the purpose of ANN search.

Like many other alternative-optimization algorithms, our algorithm is locally optimal and the final solution depends on the initialization. In the next subsection we propose a parametric solution that can be used to initialize our alternative optimization procedure.

3.2. A Parametric Solution

We further propose another solution assuming the data follows a parametric Gaussian distribution. This parametric solution has both practical and theoretical merits. First, it is a simpler method and is globally optimal if the data follows Gaussian distribution. Second, it provides a way to initialize the non-parametric method. Third, it provides new theoretical explanations for two commonly used criteria in some other quantization methods.

Algorithm 1 Non-Parametric OPQ

Input: training samples $\{\mathbf{x}\}$, number of subspaces M, number of sub-codewords k in each sub-codebook.
Output: the matrix R, sub-codebooks $\{\mathcal{C}^m\}_{m=1}^M$, M sub-indices $\{i^m\}_{m=1}^M$ for each $\mathbf{x}$.

1: Initialize R, $\{\mathcal{C}^m\}_{m=1}^M$, and $\{i^m\}_{m=1}^M$.
2: **repeat**
3: Step(i): project the data: $\hat{\mathbf{x}} = R\mathbf{x}$.
4: **for** $m = 1$ to M **do**
5: for $j = 1$ to k: update $\hat{\mathbf{c}}^m(j)$ by the sample mean of $\{\hat{\mathbf{x}}^m \mid i^m(\hat{\mathbf{x}}^m) = j\}$.
6: for $\forall \hat{\mathbf{x}}^m$: update $i^m(\hat{\mathbf{x}}^m)$ by the sub-index of the sub-codeword $\hat{\mathbf{c}}^m$ that is nearest to $\hat{\mathbf{x}}^m$.
7: **end for**
8: Step(ii): solve R by Eqn.(7).
9: **until** max iteration number reached

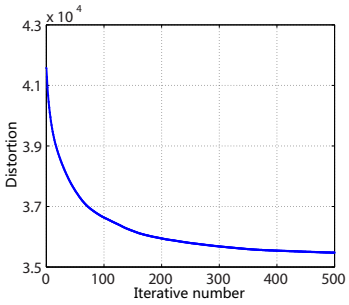

Figure 2: Convergence of Algorithm 1 in the SIFT1M dataset[10]. Here we use $M = 4$ and $k = 256$ (32 bits).

3.2.1 Distortion Bound of Quantization

We assume each dimension of $\mathbf{x} \in \mathbb{R}^D$ is subject to an independent Gaussian distribution with zero mean. From *rate distortion theory* [5] we know that the codeword length b from any quantizer achieving a distortion of E must satisfy:

$$b(E) \geq \sum_{d=1}^{D} \frac{1}{2} \log_2 \frac{D\sigma_d^2}{E}, \tag{8}$$

where σ_d^2 is the variance at each dimension. In this equation we have assumed σ_d^2 is sufficiently large (a more general form is in [5]). Equivalently, the distortion E satisfies:

$$E \geq k^{-\frac{2}{D}} D|\Sigma|^{\frac{1}{D}}, \tag{9}$$

where $k = 2^b$ and we assume all codewords have identical code length. The matrix Σ is the covariance of $\mathbf{x}$, and $|\Sigma|$

is the determinant. Here we have relaxed the independence assumption and allowed $\mathbf{x}$ to follow a Gaussian distribution $\mathcal{N}(0, \Sigma)$. Eqn.(9) is the distortion lower bound for any quantizer with k codewords. The following table shows the values of this bound and the empirical distortion of k-means (10^5 samples, $k = 256$, σ_d^2 randomly generated in $[0.5, 1]$):

D	32	64	128
distortion bound	16.2	38.8	86.7
empirical distortion	17.1	39.9	88.5

It is reasonable to consider this bound as an approximation to the k-means distortion. The small gap ($\sim 5\%$) may be due to two reasons: k-means can only achieve a locally optimal solution, and the fixed code-length for all codewords may not achieve optimal bit rate[2].

3.2.2 Distortion Bound of Product Quantization

Now we study the distortion bound of product quantization when $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$. Suppose $\mathbf{x}$ has been decomposed into a concatenation of M equal-dimensional sub-vectors. Accordingly, we can decompose Σ into $M \times M$ sub-matrices:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \cdots & \Sigma_{1M} \\ \vdots & \ddots & \vdots \\ \Sigma_{M1} & \cdots & \Sigma_{MM} \end{pmatrix}. \tag{10}$$

Here the diagonal sub-matrices Σ_{mm} are the covariance of the m-th subspace. Notice that the marginal distribution of $\mathbf{x}^m$ subjects to $\frac{D}{M}$-dimensional Gaussian $\mathcal{N}(0, \Sigma_{mm})$. From (9), the distortion bound of PQ is:

$$E_{\text{PQ}} = k^{-\frac{2M}{D}} \frac{D}{M} \sum_{m=1}^{M} |\Sigma_{mm}|^{\frac{M}{D}}, \tag{11}$$

3.2.3 Minimizing Distortion Bound of PQ

Remember that space decomposition can be parameterized by an orthonormal matrix R. When applying R to data, the variable $\hat{\mathbf{x}} = R\mathbf{x}$ is subject to $\mathcal{N}(0, \hat{\Sigma})$ with $\hat{\Sigma} = R\Sigma R^{\text{T}}$. We propose to minimize the distortion bound w.r.t. R to optimize the space decomposition in product quantization:

$$\min_{R} \sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}}, \tag{12}$$
$$s.t. \quad R^{\text{T}}R = I,$$

where $\hat{\Sigma}_{mm}$ is the diagonal sub-matrix of $\hat{\Sigma}$. The constant scale in Eqn.(11) has been ignored in this objective function. Due to the orthonormal constraint, this problem is inherently non-convex. Fortunately, the special form of our objective function can be minimized by a simple algorithm, as we show next.

[2]In information theory, it is possible to reduce the average bit rate by varying the bit-length of codewords, known as *entropy encoding* [5].

3.2.4 Eigenvalue Allocation

We first show that the objective function in Eqn.(12) has a constant lower bound. Using the *inequality of arithmetic and geometric means* (AM-GM inequality) [4], the objective in Eqn.(12) satisfies:

$$\sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}} \geq M \prod_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{1}{D}}. \tag{13}$$

The equality holds if and only if the term $|\hat{\Sigma}_{mm}|$ has the same value for all m. Further, *Fischer's inequality* [9] gives:

$$\prod_{m=1}^{M} |\hat{\Sigma}_{mm}| \geq |\hat{\Sigma}|. \tag{14}$$

The equality holds if the off-diagonal sub-matrices in $\hat{\Sigma}$ equal to zero. Note that $|\hat{\Sigma}| \equiv |\Sigma|$ is a constant given the data distribution. Combining (13) and (14), we obtain the constant lower bound for the objective in (12):

$$\sum_{m=1}^{M} |\hat{\Sigma}_{mm}|^{\frac{M}{D}} \geq M|\Sigma|^{\frac{1}{D}}. \tag{15}$$

The minimum is achieved if the following two criteria are both satisfied:

(i) Independence. If we align the data by PCA, the equality in Fischer's inequality (14) is achieved. This implies we make the dimensions independent to each other.

(ii) Balanced Subspaces' Variance. The equality in AM-GM (13) is achieved if $|\hat{\Sigma}_{mm}|$ has the same value for all subspaces. Suppose the data has been aligned by PCA. Then $|\hat{\Sigma}_{mm}|$ equals to the *product* of the eigenvalues of Σ_{mm}. By re-ordering the principal components, we can balance the product of eigenvalues for each subspace, thus the values $|\hat{\Sigma}_{mm}|$ for all subspaces. As a result, both equalities in AM-GM (13) and Fischer's (14) are satisfied, so the objective function is minimized.

Based on the above analysis, we propose a simple ***Eigenvalue Allocation*** method to optimize Eqn.(12). This method is a greedy solution for the balanced partition problem. We first align the data using PCA and sort the eigenvalues σ^2 in the descending order $\sigma_1^2 \geq ... \geq \sigma_D^2$. Note that we do not reduce dimensions. We prepare M empty buckets, one for each of the M subspaces. We sequentially pick out the largest eigenvalue and allocate it to the bucket having the minimum product of the eigenvalues in it (unless the bucket is full, *i.e.*, with D/M eigenvalues in it). The principal directions corresponding to the eigenvalues in each bucket form the subspace. In fact, the principal directions are re-ordered to form the columns of R.

In real data sets, we find this greedy algorithm is sufficiently good for minimizing the objective function. To show this fact, we compute the covariance matrix Σ from the SIFT/GIST datasets [10]. The following table shows the theoretical minimum of the objective function (right hand

side in (15)) and the objective function value (left hand side in (15)) obtained by our Eigenvalue Allocation algorithm. Here we use $M = 8$ and $k = 256$. We can see the above greedy algorithm achieves the theoretical minimum:

	theoretical minimum	Eigenvalue Allocation
SIFT	2.9286×10^3	2.9287×10^3
GIST	1.9870×10^{-3}	1.9870×10^{-3}

Interestingly, existing encoding methods have adopted, either heuristically or in objective functions different from ours, the criteria of "independence" or "balance" mentioned above. "Independence" was used in [20, 19, 3] in the form of PCA projection. "Balance" was used in [11, 20, 3]: the method in [11] rotates the data to "balance" variance for each *component* but lost "independence"; the methods in [20, 3] adaptively allocate the codeword bits to the principal components. Our derivation provides theoretical explanations for the two criteria: they can be considered as a way of minimizing the quantization distortion under a Gaussian distribution assumption.

Summary of the parametric solution. Our parametric solution first computes the covariance matrix Σ of the data and uses Eigenvalue Allocation to generate the orthonormal matrix R, which determines the space decomposition. The data are then transformed by R. The original PQ algorithm is then performed on these transformed data.

4. Experiments

We evaluate our method for ANN search using three public datasets. The first two datasets are from SIFT1M and GIST1M [10]. The SIFT1M consists of 1 million 128-d SIFT features [12] and 10k queries. The GIST1M set consists of 1 million 960-d GIST features [15] and 1k queries. The third dataset MNIST[3] consists of 70k images of handwritten digits, each as a 784-d vector concatenating all pixels. We randomly sample 1k as the queries and use the remaining as the data base. We further generate a synthetic dataset subject to a 128-d independent Gaussian distribution, where the eigenvalues of the covariance matrix are given by $\sigma_d^2 = e^{-0.1d}$ (d=1,...,128), a long-tail curve fit to the eigenvalues of the above real datasets. This synthetic set has 1 million data points and 10k queries.

We consider K=100 Euclidean nearest neighbors as the true neighbors. We find that for lookup-based methods, K is not influential for the comparisons among the methods.

We follow a common exhaustive search strategy (*e.g.*, [10, 6]). The data are ranked in the order of their approximate distances to the query. If both the query and data are to be quantized, the method is termed as *Symmetric Distance Computation* (SDC) [10]. SDC is equivalent to Hamming ranking for orthogonal hashing methods like ITQ [6]. If only the data are quantized, the method is termed as *Asymmetric Distance Computation* (ADC) [10]. ADC is more accurate than SDC but has the same complexity. We have tested both cases. The exhaustive search is fast using lookup tables: *e.g.*, for 64 bits indices it takes 20 ms per 1 million distance computation. Experiments are on a PC with an Intel Core2 2.13GHz CPU and 8G RAM. We do not combine with any non-exhaustive method like inverted files [17] as this is not the focus of our paper. We study the following methods:

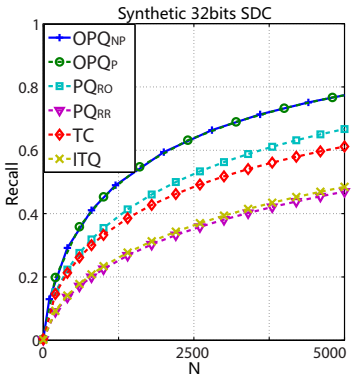

Figure 3: Comparison on Gaussian synthetic data using Symmetric Distance Computation and 32-bit codes.

- **OPQ$_{\text{P}}$**: our parametric solution.
- **OPQ$_{\text{NP}}$**: our non-parametric solution initialized by parametric solution.
- **PQ$_{\text{RO}}$**: *randomly order* dimensions as suggested in [10].
- **PQ$_{\text{RR}}$**: data is aligned using PCA and then *randomly rotated*, as suggested in [11].
- **TC** (Transform Coding [3]): a scalar quantization method that quantizes each principal component with an adaptive number of bits.
- **ITQ** [6]: one of the state-of-the-art hashing methods, a special vector quantization method.

Notice that in these settings we have assumed there is no prior knowledge available. Later we will study the case with prior knowledge.

Given the code-length B, all the PQ-based methods (OPQ$_{\text{NP}}$, OPQ$_{\text{P}}$, PQ$_{\text{RO}}$, PQ$_{\text{RR}}$) assign 8 bits to each subspace ($k = 256$). The subspace number M is $B/8$.

We have published the Matlab code of our solutions[4].

Performance on the synthetic dataset

Fig. 3 shows the performance on the synthetic data subject to a 128-d independent Gaussian distribution, evaluated through recall vs. N, *i.e.*, the proportion of the true nearest neighbors ranked in the first N positions. We can see that OPQ$_{\text{NP}}$ and OPQ$_{\text{P}}$ perform almost the same. OPQ$_{\text{P}}$ achieves

[3] http://yann.lecun.com/exdb/mnist/

[4] research.microsoft.com/en-us/um/people/kahe/

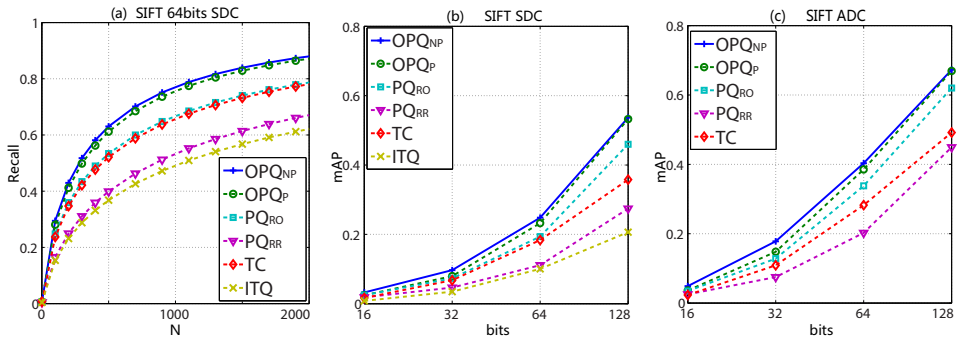

Figure 4: Comparisons on SIFT1M. (a): recall at the N top ranked samples, using SDC and 64-bit codes. (b): mean Average Precision vs. code-length, using SDC. (c): mean Average Precision vs. code-length, using ADC.

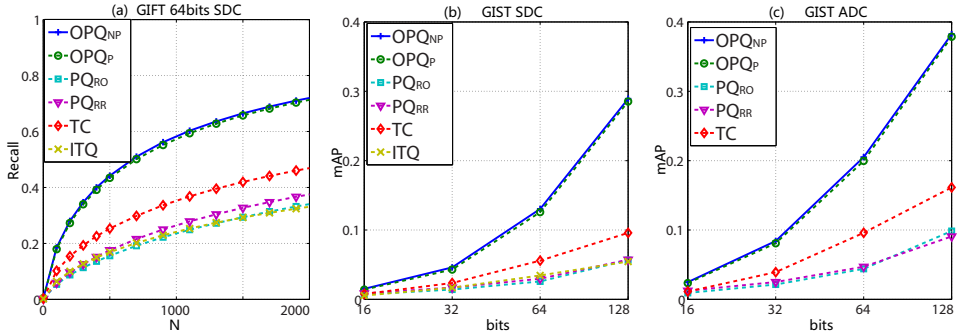

Figure 5: Comparisons on GIST1M.

theoretic minimum of 6.314×10^{-3} in Eqn.(15). This implies that, under a Gaussian distribution, our parametric solution is optimal. On the contrary, PQ_{RO} and PQ_{RR} perform substantially worse. This means that the subspace decomposition is important for PQ even under a simple Gaussian distribution. Besides, we find PQ_{RO} performs better than PQ_{RR}. This is because in this distribution PQ_{RO} automatically satisfies the independent criterion (see 3.2.4).

Performance without prior knowledge

Next we evaluate the performance if the prior knowledge is not available. In Fig. 4, 5, and 6 we compare the results of SIFT1M, GIST1M, and MNIST. We show the recall in the first N positions with 64 bits using SDC (Fig. 4, 5, 6 (a)), and the mAP (mean Average Precision) vs. code-length using SDC and ADC, respectively (Fig. 4, 5, 6 (b)(c)).

We find that both of our solutions substantially outperform the existing methods. The superiority of our methods does not depends on the choice of SDC or ADC. Typically, in all cases even our simple parametric method OPQ_{P} has shown prominent improvement over PQ_{RO} and PQ_{RR}. This indicates that PQ-based methods strongly depend on the space decomposition. We also notice the performance of PQ_{RR} is disappointing. Although this method tries to balance the variance using a random rotation, the independence between subspaces is lost by such a random rotation.

Our non-parametric solution further improves the results from the parametric one in the SIFT1M and MNIST sets. This is because these two sets exhibit more than one cluster: the SIFT1M set has two distinct clusters (this can be visualized by the first two principal components), and MNIST can be expected to have 10 clusters due to the 10 digits. In these cases the non-parametric solution is able to further reduce the distortion. Such an improvement on MNIST is significant. In GIST1M our two methods are comparable, possibly as this set is mostly subject to a Gaussian distribution.

We notice that TC performs clearly better than PQ_{RO} and PQ_{RR} in the GIST1M set. This scalar quantization method attempts to balance the bits assigned to the eigenvalues. So it better satisfies the two criteria in Sec. 3.2.4. But TC is inferior to our methods in all datasets. This is because our method quantizes multi-dimensional subspaces instead of each scalar dimension. And unlike TC that assigns a variable number of bits to each eigenvalue, our method assigns the eigenvalues to each subspace. Since bit numbers are discrete but eigenvalues are continuous, it is easier for our method to achieve balance.

Performance with prior knowledge

It has been noticed [10] that PQ works much better if utilizing the prior knowledge that SIFT and GIST are concatenated histograms. Typically, the so-called "natural" order is that each subspace consists of neighboring histograms. The "structural" order (when $M = 8$) is that each subspace consists of the same bin of all histograms (each histogram has 8 bins). It is observed [10] that the natural order perform-

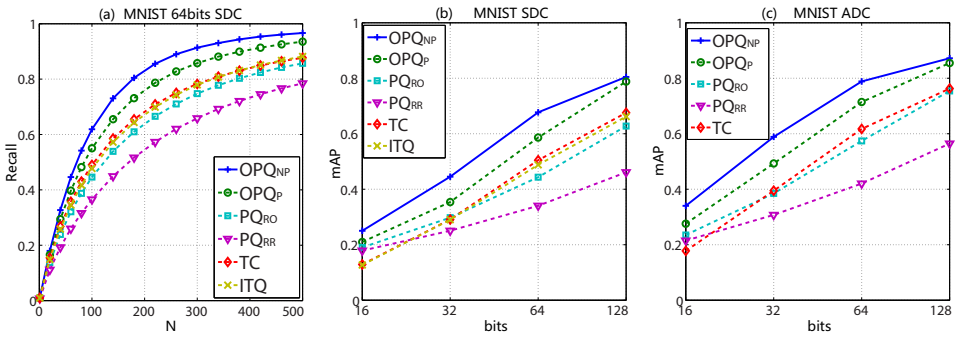

Figure 6: Comparisons on MNIST.

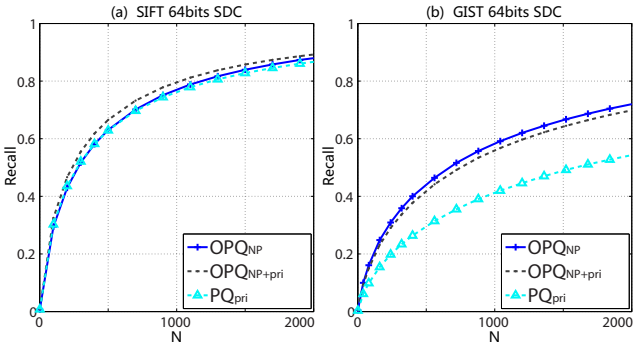

Figure 7: Comparisons with prior knowledge using 64 bits and SDC. (a): SIFT1M. (b): GIST1M.

s better for SIFT features, and the structural order is better for GIST features. We denote PQ with priori knowledge as $\mathbf{PQ}_{\text{pri}}$ (use the recommended orders). Note such priors may limit the choices of M and are not always applicable for all possible code-length.

In Fig. 7 we compare PQ_{pri} with our prior-free non-parametric method OPQ_{NP}. We also evaluate our non-parametric method using the prior orders as initialization, denoted as $\mathbf{OPQ}_{\text{NP+pri}}$. Our prior-free method outperforms the prior-based PQ. In SIFT1M our prior-dependent method improves further due to a better initialization. In GIST1M our prior-dependent method is slightly inferior than our prior-free method: this also implies that GIST1M largely follows a Gaussian distribution.

5. Conclusion

We have proposed to optimize space decomposition in product quantization. Since space decomposition has been shown to have great impact on the search accuracy in our experiments and in existing study [10], we believe our result has made product quantization a more practical and effective solution to general ANN problems.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468. IEEE, 2006.

[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[3] J. Brandt. Transform coding for fast approximate nearest neighbor search in high dimensions. In *CVPR*, 2010.

[4] A. Cauchy. *Cours d'analyse de l'École Royale Polytechnique*. Imprimerie royale, 1821.

[5] T. Cover and J. Thomas. *Elements of information theory*, chapter 13, page 348. John Wiley & Sons, Inc., 1991.

[6] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[7] R. Gray. Vector quantization. *ASSP Magazine, IEEE*, 1984.

[8] K. He, F. Wen, and J. Sun. K-means Hashing: an Affinity-Preserving Quantization Method for Learning Binary Compact Codes. In *CVPR*, 2013.

[9] R. A. Horn and C. R. Johnson. *Matrix Analysis*, chapter 7, page 478. Cambridge University Press, 1990.

[10] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33, 2011.

[11] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[13] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[14] M. Norouzi and D. Fleet. Cartesian k-means. In *CVPR*, 2013.

[15] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001.

[16] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[17] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *ICCV*, 2003.

[18] A. B. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[19] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, 2010.

[20] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.