Cartesian k-means

Mohammad Norouzi David J. Fleet
Department of Computer Science
University of Toronto
{norouzi,fleet}@cs.toronto.edu

Abstract

A fundamental limitation of quantization techniques like the k-means clustering algorithm is the storage and runtime cost associated with the large numbers of clusters required to keep quantization errors small and model fidelity high. We develop new models with a compositional parameterization of cluster centers, so representational capacity increases super-linearly in the number of parameters. This allows one to effectively quantize data using billions or trillions of centers. We formulate two such models, Orthogonal k-means *and* Cartesian k-means*. They are closely related to one another, to k-means, to methods for binary hash function optimization like ITQ [5], and to Product Quantization for vector quantization [7]. The models are tested on large-scale ANN retrieval tasks (1M GIST, 1B SIFT features), and on codebook learning for object recognition (CIFAR-10).*

1. Introduction and background

Techniques for vector quantization, like the well-known k-means algorithm, are used widely in vision and learning. Common applications include codebook learning for object recognition [16], and approximate nearest neighbor (ANN) search for information retrieval [12, 14]. In general terms, such techniques involve partitioning an input vector space into multiple regions $\{S_i\}_{i=1}^k$, mapping points $\mathbf{x}$ in each region into region-specific representatives $\{\mathbf{c}_i\}_{i=1}^k$, known as centers. As such, a quantizer, $q(\mathbf{x})$, can be expressed as

$$q(\mathbf{x}) \ = \ \sum_{i=1}^{k} 1(\mathbf{x}\!\in\! S_i)\, \mathbf{c}_i\,, \tag{1}$$

where $1(\cdot)$ is the usual indicator function.

The quality of a quantizer is expressed in terms of expected distortion, a common measure of which is squared error $\|\mathbf{x} - q(\mathbf{x})\|_2^2$. In this case, given centers $\{\mathbf{c}_i\}$, the region to which a point is assigned with minimal distortion is obtained by Euclidean nearest neighbor (NN) search. The k-means algorithm can be used to learn centers from data.

To reduce expected distortion, crucial for many applications, one can shrink region volumes by increasing k, the number of regions. In practice, however, increasing k results in prohibitive storage and run-time costs. Even if one resorts to ANN search with approximate k-means [14] or hierarchical k-means [12], it is hard to scale to large k (*e.g.,* $k = 2^{64}$), as storing the centers is untenable.

This paper concerns methods for scalable quantization with tractable storage and run-time costs. Inspired by Product Quantization (PQ), a state-of-the-art algorithm for ANN search with high-dimensional data (*e.g.,* [7]), compositionality is one key. By expressing data in terms of recurring, reusable parts, the representational capacity of compositional models grows exponentially in the number of parameters. Compression techniques like JPEG accomplish this by encoding images as disjoint rectangular patches. PQ divides the feature space into disjoint subspaces that are quantized independently. Other examples include part-based recognition models (*e.g.,* [18]), and tensor-based models for style-content separation (*e.g.,* [17]). Here, with a compositional parameterization of region centers, we find a family of models that reduce the encoding cost of k centers down from k to between $\log_2 k$ and $\sqrt{k}$. A model parameter controls the trade-off between model fidelity and compactness.

We formulate two related algorithms, *Orthogonal k-means* (ok-means) and *Cartesian k-means* (ck-means). They are natural extensions of k-means, and are closely related to other hashing and quantization methods. The ok-means algorithm is a generalization of the Iterative Quantization (ITQ) algorithm for finding locality-sensitive binary codes [5]. The ck-means model is an extension of ok-means, and can be viewed as a generalization of PQ.[1]

We then report empirical results on large-scale ANN search tasks, and on codebook learning for recognition. For ANN search we use datasets of *1M GIST* and *1B SIFT* features, with both symmetric and asymmetric distance measures on the latent codes. We consider codebook learning for a generic form of recognition on CIFAR-10.

2. k-means

Given a dataset of n p-dim points, $\mathcal{D} \equiv \{\mathbf{x}_j\}_{j=1}^n$, the k-means algorithm partitions the n points into k clusters, and

[1] A very similar generalization of PQ, was developed concurrently by Ge, He, Ke and Sun, and also appears in CVPR 2013 [4].

represents each cluster by a center point. Let $C \in \mathbb{R}^{p\times k}$ be a matrix whose columns comprise the k cluster centers, *i.e.,* $C = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_k]$. The k-means objective is to minimize the within-cluster squared distances:

$$\begin{aligned} \ell_{\text{k-means}}(C) &= \sum_{\mathbf{x}\in\mathcal{D}} \min_i \|\mathbf{x} - \mathbf{c}_i\|_2^2 \\ &= \sum_{\mathbf{x}\in\mathcal{D}} \min_{\mathbf{b}\in\mathcal{H}_{1/k}} \|\mathbf{x} - C\mathbf{b}\|_2^2 \end{aligned} \qquad (2)$$

where $\mathcal{H}_{1/k} \equiv \{\mathbf{b} \mid \mathbf{b} \in \{0,1\}^k \text{ and } \|\mathbf{b}\| = 1\}$, *i.e.,* $\mathbf{b}$ is a binary vector comprising a 1-of-k encoding. Lloyd's k-means algorithm [11] finds a local minimum of (2) by iterative, alternating optimization with respect to C and the $\mathbf{b}$'s.

The k-means model is simple and intuitive, using NN search to assign points to centers. The assignment of points to centers can be represented with a $\log k$-bit index per data point. The cost of storing the centers grows linearly with k.

3. Orthogonal k-means with 2^m centers

With a compositional model one can represent cluster centers more efficiently. One such approach is to reconstruct each input with an additive combination of the columns of C. To this end, instead of the 1-of-k encoding in (2), we let $\mathbf{b}$ be a general m-bit vector, $\mathbf{b} \in \mathcal{H}_m \equiv \{0,1\}^m$, and $C \in \mathbb{R}^{p\times m}$. As such, each cluster center is the sum of a subset of the columns of C. There are 2^m possible subsets, and therefore $k = 2^m$ centers. While the number of parameters in the quantizer is linear in m, the number of centers increases exponentially.

While efficient in representing cluster centers, the approach is problematic, because solving

$$\hat{\mathbf{b}} = \operatorname*{argmin}_{\mathbf{b}\in\mathcal{H}_m} \|\mathbf{x} - C\mathbf{b}\|_2^2\,, \qquad (3)$$

is intractable; *i.e.,* the discrete optimization is not submodular. Obviously, for small 2^m one could generate all possible centers and then perform NN search to find the optimal solution, but this would not scale well to large values of m.

One key observation is that if the columns of C are orthogonal, then optimization (3) becomes tractable. To explain this, without loss of generality, assume the bits belong to $\{-1, 1\}$ instead of $\{0, 1\}$, *i.e.,* $\mathbf{b}' \in \mathcal{H}'_m \equiv \{-1,1\}^m$. Then,

$$\|\mathbf{x} - C\mathbf{b}'\|_2^2 = \mathbf{x}^\mathsf{T}\mathbf{x} + \mathbf{b}'^\mathsf{T} C^\mathsf{T} C\mathbf{b}' - 2\mathbf{x}^\mathsf{T} C\mathbf{b}'\,. \qquad (4)$$

For diagonal $C^\mathsf{T}C$, the middle quadratic term on the RHS becomes $\mathrm{trace}(C^\mathsf{T}C)$, independent of $\mathbf{b}'$. As a consequence, when C has orthogonal columns,

$$\operatorname*{argmin}_{\mathbf{b}'\in\mathcal{H}'_m} \|\,\mathbf{x} - C\mathbf{b}'\,\|_2^2 = \mathrm{sgn}(C^\mathsf{T}\mathbf{x})\,, \qquad (5)$$

where $\mathrm{sgn}(\cdot)$ is the element-wise sign function.

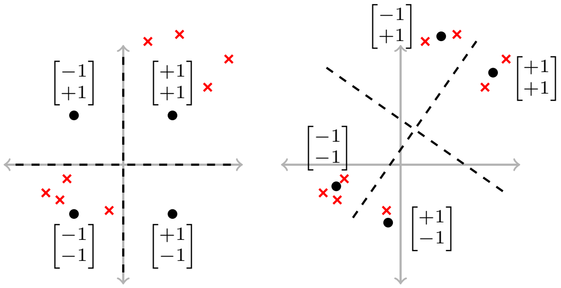
$\begin{bmatrix}-1\\+1\end{bmatrix}$
$\begin{bmatrix}+1\\+1\end{bmatrix}$
$\begin{bmatrix}-1\\-1\end{bmatrix}$
$\begin{bmatrix}+1\\-1\end{bmatrix}$
$\begin{bmatrix}-1\\+1\end{bmatrix}$
$\begin{bmatrix}+1\\+1\end{bmatrix}$
$\begin{bmatrix}-1\\-1\end{bmatrix}$
$\begin{bmatrix}+1\\-1\end{bmatrix}$

Figure 1. Two quantizations of 2D data (red ×'s) by ok-means. Cluster centers are depicted by dots, and cluster boundaries by dashed lines. (Left) Clusters formed by a 2-cube with no rotation, scaling or translation; centers = $\{\mathbf{b}' | \mathbf{b}' \in \mathcal{H}'_2\}$. (Right) Rotation, scaling and translation are used to reduce distances between data points and cluster centers; centers = $\{\mu + RD\mathbf{b}' \mid \mathbf{b}' \in \mathcal{H}'_2\}$.

To reduce quantization error further we can also introduce an offset, denoted μ, to translate $\mathbf{x}$. Taken together with (5), this leads to the loss function for the model we call *orthogonal k-means* (ok-means):[2]

$$\ell_{\text{ok-means}}(C, \mu) = \sum_{\mathbf{x}\in\mathcal{D}} \min_{\mathbf{b}'\in\mathcal{H}'_m} \|\mathbf{x} - \mu - C\mathbf{b}'\|_2^2\,. \qquad (6)$$

Clearly, with a change of variables, $\mathbf{b}' = 2\mathbf{b} - \mathbf{1}$, we can define new versions of μ and C, with identical loss, for which the unknowns are binary, but in $\{0, 1\}^m$.

The ok-means quantizer encodes each data point as a vertex of a transformed m-dimensional unit hypercube. The transform, via C and μ, maps the hypercube vertices onto the input feature space, ideally as close as possible to the data points. The matrix C has orthogonal columns and can therefore be expressed in terms of rotation and scaling; *i.e.,* $C \equiv RD$, where $R \in \mathbb{R}^{p\times m}$ has orthonormal columns ($R^\mathsf{T}R = \mathbb{I}_m$), and D is diagonal and positive definite. The goal of learning is to find the parameters, R, D, and μ, which minimize quantization error.

Fig. 1 depicts a small set of 2D data points (red x's) and two possible quantizations. Fig. 1 (left) depicts the vertices of a 2-cube with $C = \mathbb{I}_2$ and zero translation. The cluster regions are simply the four quadrants of the 2D space. The distances between data points and cluster centers, *i.e.,* the quantization errors, are relatively large. By comparison, Fig. 1 (right) shows how a transformed 2-cube, the full model, can greatly reduce quantization errors.

3.1. Learning ok-means

To derive the learning algorithm for ok-means we first rewrite the objective in matrix terms. Given n data points, let $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$. Let $B' \in \{-1, 1\}^{m\times n}$ denote the corresponding cluster assignment coefficients. Our

[2] While closely related to ITQ, we use the term ok-means to emphasize the relationship to k-means.

goal is to find the assignment coefficients B' and the transformation parameters, namely, the rotation R, scaling D, and translation μ, that minimize

$$\ell_{\text{ok-means}}(B', R, D, \mu) = \|X - \mu \mathbf{1}^\mathsf{T} - RDB'\|_\mathrm{f}^2 \quad (7)$$
$$= \|X' - RDB'\|_\mathrm{f}^2 \quad (8)$$

where $\|\cdot\|_\mathrm{f}$ denotes the Frobenius norm, $X' \equiv X - \mu \mathbf{1}^\mathsf{T}$, R is constrained to have orthonormal columns ($R^\mathsf{T} R = \mathbb{I}_m$), and D is a positive diagonal matrix.

Like k-means, coordinate descent is effective for optimizing (8). We first initialize μ and R, and then iteratively optimize $\ell_{\text{ok-means}}$ with respect to B', D, R, and μ:

- *Optimize B' and D*: With straightforward algebraic manipulation, one can show that

  $$\ell_{\text{ok-means}} = \|R^\mathsf{T} X' - DB'\|_\mathrm{f}^2 + \|{R^\perp}^\mathsf{T} X'\|_\mathrm{f}^2 \,, \quad (9)$$

  where columns of $R^\perp$ span the orthogonal complement of the column-space of R (*i.e.,* the block matrix $[R \;\; R^\perp]$ is orthogonal).

  It follows that, given X' and R, we can optimize the first term in (9) to solve for B' and D. Here, DB' is the least-squares approximation to $R^\mathsf{T} X'$, where $R^\mathsf{T} X'$ and DB' are $m \times n$. Further, the i^{th} row of DB' can only contain elements from $\{-d_i, +d_i\}$ where $d_i = D_{ii}$. Wherever the corresponding element of $R^\mathsf{T} X'$ is positive (negative) we should put a positive (negative) value in DB'. The optimal d_i is determined by the mean absolute value of the elements in the i^{th} row of $R^\mathsf{T} X'$:

  $$B' \;\leftarrow\; \operatorname{sgn}\left(R^\mathsf{T} X'\right) \quad (10)$$
  $$D \;\leftarrow\; \operatorname{Diag}(\underset{\text{row}}{\operatorname{mean}}(\operatorname{abs}(R^\mathsf{T} X'))) \quad (11)$$

- *Optimize R*: Using the original objective (8), find R that minimizes $\|X' - RA\|_\mathrm{f}^2$, subject to $R^\mathsf{T} R = \mathbb{I}_m$, and $A \equiv DB'$. This is equivalent to an Orthogonal Procrustes problem [15], and can be solved exactly using SVD. In particular, by adding $p - m$ rows of zeros to the bottom of D, DB becomes $p \times n$. Then R is square and orthogonal and can be estimated with SVD. But since DB is degenerate we are only interested in the first m columns of R; the remaining columns are unique only up to rotation of the null-space.)
- *Optimize μ*: Given R, B' and D, the optimal μ is given by the column average of $X - RDB'$:

  $$\mu \;\leftarrow\; \underset{\text{colum}}{\operatorname{mean}}(X - RDB') \quad (12)$$

3.2. Approximate nearest neighbor search

One application of scalable quantization is ANN search. Before introducing more advanced quantization techniques, we describe some experimental results with ok-means on Euclidean ANN search. The benefits of ok-means for ANN search are two-fold. Storage costs for the centers is reduced to $O(\log k)$, from $O(k)$ with k-means. Second, substantial speedups are possible by exploiting fast methods for NN search on binary codes in Hamming space (*e.g.,* [13]).

Generally, in terms of a generic quantizer $q(\cdot)$, there are two natural ways to estimate the distance between two vectors, $\mathbf{v}$ and $\mathbf{u}$ [7]. Using the *Symmetric quantizer distance (SQD)* $\|\mathbf{v}{-}\mathbf{u}\|$ is approximated by $\|q(\mathbf{v}){-}q(\mathbf{u})\|$. Using the *Asymmetric quantizer distance (AQD)*, only one of the two vectors is quantized, and $\|\mathbf{v}{-}\mathbf{u}\|$ is estimated as $\|\mathbf{v}{-}q(\mathbf{u})\|$. While SQD might be slightly faster to compute, AQD incurs lower quantization errors, and hence is more accurate.

For ANN search, in a pre-processing stage, each database entry, $\mathbf{u}$, is encoded into a binary vector corresponding to the cluster center index to which $\mathbf{u}$ is assigned. At test time, the queries may or may not be encoded into indices, depending on whether one uses SQD or AQD.

In the ok-means model, the quantization of an input $\mathbf{x}$ is straightforwardly shown to be

$$q_{\text{ok}}(\mathbf{x}) = \mu + RD \operatorname{sgn}(R^\mathsf{T}(\mathbf{x} - \mu)) \,. \quad (13)$$

The corresponding m-bit cluster index is $\operatorname{sgn}(R^\mathsf{T}(\mathbf{x} - \mu))$. Given two indices, namely $\mathbf{b}'_1, \mathbf{b}'_2 \in \{-1, +1\}^m$, the symmetric ok-means quantizer distance is

$$SQD_{\text{ok}}(\mathbf{b}'_1, \mathbf{b}'_2) = \|\mu + RD\mathbf{b}'_1 - \mu - RD\mathbf{b}'_2\|_2^2$$
$$= \|D(\mathbf{b}'_1 - \mathbf{b}'_2)\|_2^2 \,. \quad (14)$$

In effect, SQD_{ok} is a weighted Hamming distance. It is the sum of the squared diagonal entries of D corresponding to bits where $\mathbf{b}'_1$ and $\mathbf{b}'_2$ differ. Interestingly, in our experiments with ok-means, Hamming and weighted Hamming distances yield similar results. Thus, in ok-means experiments we simply report results for Hamming distance, to facilitate comparisons with other techniques. When the scaling in ok-means is constrained to be isotropic (*i.e.,* $D = \alpha \mathbb{I}_m$ for $\alpha \in \mathbb{R}^+$), then SQD_{ok} becomes a constant multiple of the usual Hamming distance. As discussed in Sec. 5, this isotropic ok-means is closely related to ITQ [5].

The ok-means AQD between a feature vector $\mathbf{x}$ and a cluster index $\mathbf{b}'$, is defined as

$$AQD_{\text{ok}}(\mathbf{x}, \mathbf{b}') = \|\mathbf{x} - \mu - RD\mathbf{b}'\|_2^2$$
$$= \|R^\mathsf{T}\mathbf{x}' - D\mathbf{b}'\|_2^2 + \|{R^\perp}^\mathsf{T}\mathbf{x}'\|_2^2 \,, \quad (15)$$

where $\mathbf{x}' = \mathbf{x}{-}\mu$. For ANN search, in comparing distances from query $\mathbf{x}$ to a dataset of binary indices, the second term on the RHS of (15) is irrelevant, since it does not depend on $\mathbf{b}'$. Without this term, AQD_{ok} becomes a form of asymmetric Hamming (AH) distance between $R^\mathsf{T}\mathbf{x}'$ and $\mathbf{b}'$. While previous work [6] discussed various *ad hoc* AH distance measures for binary hashing, interestingly, the optimal AH distance for ok-means is derived directly in our model.

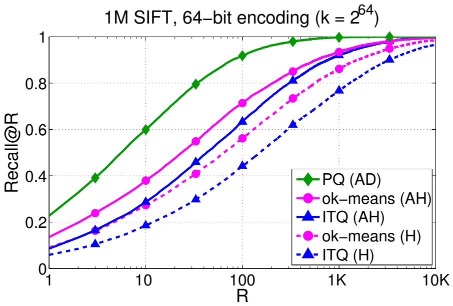

Figure 2. Euclidean ANN retrieval results for different methods and distance functions on the 1M SIFT dataset.

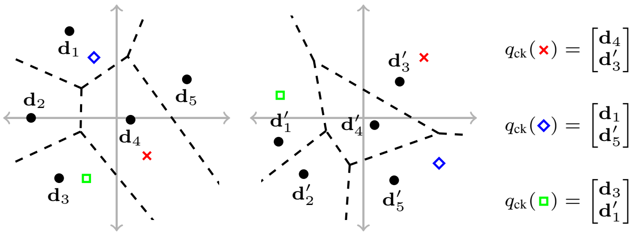
$\mathbf{d}_1$
$\mathbf{d}_2$
$\mathbf{d}_3$
$\mathbf{d}_4$
$\mathbf{d}_5$
$\mathbf{d}'_1$
$\mathbf{d}'_2$
$\mathbf{d}'_3$
$\mathbf{d}'_4$
$\mathbf{d}'_5$
$q_{\text{ck}}(×) = \begin{bmatrix}\mathbf{d}_4\\\mathbf{d}'_3\end{bmatrix}$
$q_{\text{ck}}(◇) = \begin{bmatrix}\mathbf{d}_1\\\mathbf{d}'_5\end{bmatrix}$
$q_{\text{ck}}(□) = \begin{bmatrix}\mathbf{d}_3\\\mathbf{d}'_1\end{bmatrix}$

Figure 3. Depiction of Cartesian quantization on 4D data, with the first (last) two dimensions sub-quantized on the left (right). Cartesian k-means quantizer denoted q_{ck}, combines the sub-quantizations in subspaces, and produces a 4D reconstruction.

3.3. Experiments with ok-means

Following [7], we report ANN search results on *1M SIFT*, a corpus of 128D SIFT descriptors with disjoint sets of 100K training, 1M base, and 10K test vectors. The training set is used to train models. The base set is the database, and the test points are queries. The number of bits m is typically less than p, but no pre-processing is needed for dimensionality reduction. Rather, to initialize learning, R is a random rotation of the first m principal directions of the training data, and μ is the mean of the data.

For each query we find R nearest neighbors, and compute *Recall@R*, the fraction of queries for which the ground-truth Euclidean NN is found in the R retrieved items. Fig. 2 shows the Recall@R plots for ok-means with Hamming $(H) \approx SQD_{\text{ok}}$ and asymmetric Hamming (AH) $\equiv$ AQD_{ok} distance, *vs.* ITQ [5] and PQ [7]. The PQ method exploits a more complex asymmetric distance function denoted $AD \equiv AQD_{\text{ck}}$ (defined in Sec. 6.1). Note first that ok-means improves upon ITQ, with both Hamming and asymmetric Hamming distances. This is due to the scaling parameters (*i.e., D*) in ok-means. If one is interested in Hamming distance efficient retrieval, ok-means is prefered over ITQ. But better results are obtained with the asymmetric distance function.

Fig. 2 also shows that PQ achieves superior recall rates. This stems from its joint encoding of multiple feature dimensions. In ok-means, each bit represents a partition of the feature space into two clusters, separated by a hyperplane. The intersection of m orthogonal hyperplanes yields 2^m regions. Hence we obtain just two clusters per dimension, and each dimension is encoded independently. In PQ, by comparison, multiple dimensions are encoded jointly, with arbitrary numbers of clusters. PQ thereby captures statistical dependencies among different dimensions. We next extend ok-means to jointly encode multiple dimensions.

4. Cartesian k-means

In the *Cartesian k-means* (ck-means) model, each region center is expressed parametrically as an additive combination of multiple *subcenters*. Let there be m sets of subcenters, each with h elements.[3] Let $C^{(i)}$ be a matrix whose columns comprise the elements of the i^{th} subcenter set; $C^{(i)} \in \mathbb{R}^{p \times h}$. Finally, assume that each cluster center, $\mathbf{c}$, is the sum of exactly one element from each subcenter set:

$$\mathbf{c} \;=\; \sum_{i=1}^{m} C^{(i)}\mathbf{b}^{(i)}\,, \tag{16}$$

where $\mathbf{b}^{(i)} \in \mathcal{H}_{1/h}$ is a 1-of-h encoding.

As a concrete example (see Fig. 3), suppose we are given 4D inputs, $\mathbf{x} \in \mathbb{R}^4$, and we split each datum into $m = 2$ parts:

$$\mathbf{z}^{(1)} = \begin{bmatrix}\mathbb{I}_2 & 0\end{bmatrix}\mathbf{x}\,, \quad \text{and} \quad \mathbf{z}^{(2)} = \begin{bmatrix}0 & \mathbb{I}_2\end{bmatrix}\mathbf{x}\,. \tag{17}$$

Then, suppose we quantize each part, $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$, separately. As depicted in Fig. 3 (left and middle), we could use $h = 5$ subcenters for each one. Placing the corresponding subcenters in the columns of 4×5 matrices $C^{(1)}$ and $C^{(2)}$,

$$C^{(1)} = \begin{bmatrix}\mathbf{d}_1 \; \mathbf{d}_2 \; \mathbf{d}_3 \; \mathbf{d}_4 \; \mathbf{d}_5\\ \mathbf{0}_{2\times5}\end{bmatrix}, \quad C^{(2)} = \begin{bmatrix}\mathbf{0}_{2\times5}\\ \mathbf{d}'_1 \; \mathbf{d}'_2 \; \mathbf{d}'_3 \; \mathbf{d}'_4 \; \mathbf{d}'_5\end{bmatrix},$$

we obtain a model (16) that provides 5^2 possible centers with which to quantize the data.

More generally, the total number of model centers is $k = h^m$. Each center is a member of the Cartesian product of the subcenter sets, hence the name *Cartesian k-means*. Importantly, while the number of centers is h^m, the number of subcenters is only mh. The model provides a super-linear number of centers with a linear number of parameters.

The learning objective for Cartesian k-means is

$$\ell_{\text{ck-means}}(C) \;=\; \sum_{\mathbf{x}\in\mathcal{D}} \min_{\{\mathbf{b}^{(i)}\}_{i=1}^{m}} \left\|\mathbf{x} - \sum_{i=1}^{m} C^{(i)}\mathbf{b}^{(i)}\right\|_2^2 \tag{18}$$

where $\mathbf{b}^{(i)} \in \mathcal{H}_{1/h}$, and $C \equiv [C^{(1)}, \cdots, C^{(m)}] \in \mathbb{R}^{p \times mh}$. If we let $\mathbf{b}^{\mathsf{T}} \equiv [{\mathbf{b}^{(1)}}^{\mathsf{T}}, \cdots, {\mathbf{b}^{(m)}}^{\mathsf{T}}]$ then the second sum in (18) can be expressed succinctly as $C\mathbf{b}$.

[3] While here we assume a fixed cardinality for all subcenter sets, the model is easily extended to allow sets with different cardinalities.

The key problem with this formulation is that the minimization of (18) with respect to the $\mathbf{b}^{(i)}$'s is intractable. Nevertheless, motivated by orthogonal k-means (Sec. 3), encoding can be shown to be both efficient and exact if we impose orthogonality constraints on the sets of subcenters. To that end, assume that all subcenters in different sets are pairwise orthogonal; *i.e.,*

$$\forall i, j \mid i \neq j \quad {C^{(i)}}^{\mathsf{T}} C^{(j)} \;=\; 0_{h\times h}\,. \tag{19}$$

Each subcenter matrix $C^{(i)}$ spans a linear subspace of $\mathbb{R}^p$, and the linear subspaces for different subcenter sets do not intersect. Hence, (19) implies that only the subcenters in $C^{(i)}$ can explain the projection of $\mathbf{x}$ onto the $C^{(i)}$ subspace.

In the example depicted in Fig. 3, the input features are simply partitioned (17), and the subspaces clearly satisfy the orthogonality constraints. It is also clear that $C \equiv \left[\, C^{(1)} \;\; C^{(2)} \right]$ is block diagonal, with 2×5 blocks, denoted $D^{(1)}$ and $D^{(2)}$. The quantization error therefore becomes

$$\begin{aligned}\left\|\mathbf{x} - C\mathbf{b}\right\|_2^2 \;&=\; \left\| \begin{bmatrix}\mathbf{z}^{(1)}\\ \mathbf{z}^{(2)}\end{bmatrix} - \begin{bmatrix}D^{(1)} & 0\\ 0 & D^{(2)}\end{bmatrix} \begin{bmatrix}\mathbf{b}^{(1)}\\ \mathbf{b}^{(2)}\end{bmatrix} \right\|^2 \\ &=\; \left\|\mathbf{z}^{(1)} - D^{(1)}\mathbf{b}^{(1)}\right\|^2 + \left\|\mathbf{z}^{(2)} - D^{(2)}\mathbf{b}^{(2)}\right\|^2 .\end{aligned}$$

In words, the squared quantization error is the sum of the squared errors on the subspaces. One can therefore solve for the binary coefficients of the subcenter sets independently.

In the general case, assuming (19) is satisfied, C can be expressed as a product $R\,D$, where R has orthonormal columns, and D is block diagonal; *i.e.,* $C = R\,D$ where

$$R = [R^{(1)}, \cdots, R^{(m)}]\,, \;\text{ and }\; D = \begin{bmatrix} D^{(1)} & 0 & \dots & 0\\ 0 & D^{(2)} & & 0\\ \vdots & & \ddots & \vdots\\ 0 & 0 & \dots & D^{(m)}\end{bmatrix}, \tag{20}$$

and hence $C^{(i)} = R^{(i)} D^{(i)}$. With $s_i \equiv \operatorname{rank}(C^{(i)})$, it follows that $D^{(i)} \in \mathbb{R}^{s_i \times h}$ and $R^{(i)} \in \mathbb{R}^{p\times s_i}$. Clearly, $\sum s_i \leq p$, because of the orthogonality constraints.

Replacing $C^{(i)}$ with $R^{(i)} D^{(i)}$ in the RHS of (18), we find

$$\left\|\mathbf{x} - C\mathbf{b}\right\|_2^2 \;=\; \sum_{i=1}^{m} \left\|\mathbf{z}^{(i)} - D^{(i)}\mathbf{b}^{(i)}\right\|_2^2 + \|{R^{\perp}}^{\mathsf{T}}\mathbf{x}\|_2^2\,, \tag{21}$$

where $\mathbf{z}^{(i)} \equiv {R^{(i)}}^{\mathsf{T}}\mathbf{x}$, and $R^{\perp}$ is the orthogonal complement of R. This shows that, with orthogonality constraints (19), the ck-means encoding problem can be split into m independent sub-encoding problems, one for each subcenter set.

To find the $\mathbf{b}^{(i)}$ that minimizes $\left\|\mathbf{z}^{(i)} - D^{(i)}\mathbf{b}^{(i)}\right\|_2^2$, we perform NN search with $\mathbf{z}^{(i)}$ against h s_i-dim vectors in $D^{(i)}$. This entails a cost of $O(hs_i)$. Fortunately, all the elements of $\mathbf{b}$ can be found very efficiently, in $O(hs)$, where $s \equiv \sum s_i$. If we also include the cost of rotating $\mathbf{x}$ to obtain each $\mathbf{z}^{(i)}$, the total encoding cost is $O(ps + hs)$, *i.e.,* $O(p^2{+}hp)$. Alternatively, one could perform NN search in p-dim $C^{(i)}$'s, to find the $\mathbf{b}^{(i)}$'s, which costs $O(mhp)$. Table 1 summerizes the models in terms of their number of centers, index size, and cost of storage and encoding.

method	#centers	#bits	cost	cost(s)
ok-means	2^m	m	$O(mp)$	$O(mp)$
ck-means	h^m	$m \log h$	$O(p^2 + hp)$ or $O(mhp)$	$O(ps{+}hs)$ or $O(ps{+}mhs)$
k-means	k	$\log k$	$O(kp)$	$O(ps + ks)$

Table 1. A summery of ok-means, ck-means, and k-means in terms of number of centers, number of bits needed for indices (*i.e.,* log #centers), and the storage cost of representation, which is the same as the encoding cost to convert inputs to indices. The last column shows the costs given an assumption that C has a rank of $s \geq m$.

4.1. Learning ck-means

We can re-write the ck-means objective (18) in matrix form with the Frobenius norm; *i.e.,*

$$\ell_{\text{ck-means}}(R, D, B) \;=\; \|\, X - RDB\,\|_{\text{f}}^2 \tag{22}$$

where the columns of X and B comprise the data points and the subcenter assignment coefficients. The input to the learning algorithm is the training data X, the number of subcenter sets m, the cardinality of the subcenter sets h, and an upper bound on the rank of C, *i.e.,* s. In practice, we also let each rotation matrix $R^{(i)}$ have the same number of columns, *i.e.,* $s_i = s/m$. The outputs are the matrices $\{R^{(i)}\}$ and $\{D^{(i)}\}$ that provide a local minima of (22).

Learning begins with the initialization of R and D, followed by iterative coordinate descent in B, D, and R:

- *Optimize B and D*: With R fixed, the objective is given by (21) where $\|{R^{\perp}}^{\mathsf{T}} X\|_{\text{f}}^2$ is constant. Given data projections $Z^{(i)} \equiv {R^{(i)}}^{\mathsf{T}} X$, to optimize for B and D we perform one step of k-means for each subcenter set:
  - *Assignment:* Perform NN searches for each subcenter set to find the assignment coefficients, $B^{(i)}$,
  $$B^{(i)} \leftarrow \operatorname*{argmin}_{B^{(i)}} \|Z^{(i)} - D^{(i)}B^{(i)}\|_{\text{f}}^2$$
  - *Update:* $$D^{(i)} \leftarrow \operatorname*{argmin}_{D^{(i)}} \|Z^{(i)} - D^{(i)}B^{(i)}\|_{\text{f}}^2$$
- *Optimize R*: Placing the $D^{(i)}$'s along the diagonal of D, as in (20), and concatenating $B^{(i)}$'s as rows of B, *i.e.,* $B^{\mathsf{T}} = [{B^{(1)}}^{\mathsf{T}}, \ldots, {B^{(m)}}^{\mathsf{T}}]$, the optimization of R reduces to the orthogonal Procrustes problem:
$$R \leftarrow \operatorname*{argmin}_{R} \|X - RDB\|_{\text{f}}^2\,.$$

In experiments below, $R \in \mathbb{R}^{p\times p}$, and $\operatorname{rank}(C) \leq p$ is unconstrained. For high-dimensional data where $\operatorname{rank}(X) \ll p$, for efficiency it may be useful to constrain $\operatorname{rank}(C)$.

5. Relations and related work

As mentioned above, there are close mathematical relationships between ok-means, ck-means, ITQ for binary hashing [5], and PQ for vector quantization [7]. It is instructive to specify these relationships in some detail.

Iterative Quantization *vs.* Orthogonal k-means. ITQ [5] is a variant of locality-sensitive hashing, mapping data to binary codes for fast retrieval. To extract m-bit codes, ITQ first zero-centers the data matrix to obtain X'. PCA is then used for dimensionality reduction, from p down to m dimensions, after which the subspace representation is randomly rotated. The composition of PCA and the random rotation can be expressed as $W^\mathsf{T} X'$ where $W \in \mathbb{R}^{p\times m}$. ITQ then solves for the $m{\times}m$ rotation matrix, R, that minimizes

$$\ell_{\text{ITQ}}(B', R) \;=\; \|R^\mathsf{T} W^\mathsf{T} X' - B'\|_\text{f}^2\,, \;\text{ s.t. }\; R^\mathsf{T} R = \mathbb{I}_{m\times m}\,, \tag{23}$$

where $B' \in \{-1, +1\}^{n\times p}$.

ITQ rotates the subspace representation of the data to match the binary codes, and then minimizes quantization error within the subspace. By contrast, ok-means maps the binary codes into the original input space, and then considers both the quantization error within the subspace and the out-of-subspace projection error. A key difference is the inclusion of $\|R^\perp X'\|_\text{f}^2$ in the ok-means objective (9). This is important since one can often reduce quantization errors by projecting out significant portions of the feature space.

Another key difference between ITQ and ok-means is the inclusion of non-uniform scaling in ok-means. This is important when the data are not isotropic, and contributes to the marked improvement in recall rates in Fig. 2.

Orthogonal k-means *vs.* Cartesian k-means. We next show that ok-means is a special case of ck-means with $h = 2$, where each subcenter set has two elements. To this end, let $C^{(i)} = [\mathbf{c}_1^{(i)}\; \mathbf{c}_2^{(i)}]$, and let $\mathbf{b}^{(i)} = [b_1^{(i)}\; b_2^{(i)}]^\mathsf{T}$ be the i^{th} subcenter matrix selection vector. Since $\mathbf{b}^{(i)}$ is a 1-of-2 encoding ($\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ or $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$), it follows that:

$$b_1^{(i)}\mathbf{c}_1^{(i)} + b_2^{(i)}\mathbf{c}_2^{(i)} \;=\; \frac{\mathbf{c}_1^{(i)} + \mathbf{c}_2^{(i)}}{2} + b'_i\,\frac{\mathbf{c}_1^{(i)} - \mathbf{c}_2^{(i)}}{2}\,, \tag{24}$$

where $b'_i \;\equiv\; b_1^{(i)} - b_2^{(i)} \in \{-1, +1\}$. With the following setting of the ok-means parameters,

$$\mu = \sum_{i=1}^{m} \frac{\mathbf{c}_1^{(i)} + \mathbf{c}_2^{(i)}}{2}, \text{ and } C = \left[\frac{\mathbf{c}_1^{(1)} - \mathbf{c}_2^{(1)}}{2}, \ldots, \frac{\mathbf{c}_1^{(m)} - \mathbf{c}_2^{(m)}}{2}\right],$$

it should be clear that $\sum_i C^{(i)}\mathbf{b}^{(i)} \;=\; \mu + C\mathbf{b}'$, where $\mathbf{b}' \in \{-1, +1\}^m$, and b'_i is the i^{th} bit of $\mathbf{b}'$, used in (24). Similarly, one can also map ok-means parameters onto corresponding subcenters for ck-means.

Thus, there is a 1-to-1 mapping between the parameterizations of cluster centers in ok-means and ck-means for $h = 2$. The benefits of ok-means are its small number of parameters, and its intuitive formulation. The benefit of the ck-means generalization is its joint encoding of multiple dimensions with an arbitrary number of centers, allowing it to capture intrinsic dependence among data dimensions.

Product Quantization *vs.* Cartesian k-means. PQ first splits the input vectors into m disjoint sub-vectors, and then quantizes each sub-vector separately using a sub-quantizer. Thus PQ is a special case of ck-means where the rotation R is not optimized; rather, R is fixed in both learning and retrieval. This is important because the sub-quantizers act independently, thereby ignoring intra-subspace statistical dependence. Thus the selection of subspaces is critical for PQ [7, 8]. Jégou et al. [8] suggest using PCA, followed by random rotation, before applying PQ to high-dimensional vectors. Finding the rotation to minimize quantization error is mentioned in [8], but it was considered too difficult to estimate.

Here we show that one can find a rotation to minimize the quantization error. The ck-means learning algorithm optimizes sub-quantizers in the inner loop, but they interact in an outer loop that optimizes the rotation (Sec. 4.1).

6. Experiments

6.1. Euclidean ANN search

Euclidean ANN is a useful task for comparing quantization techniques. Given a database of ck-means indices, and a query, we use Asymmetric and Symmetric ck-means quantizer distance, denoted AQD_{ck} and SQD_{ck}. The AQD_{ck} between a query $\mathbf{x}$ and a binary index $\mathbf{b} \equiv \left[{b^{(1)}}^\mathsf{T}, \ldots, {b^{(m)}}^\mathsf{T}\right]^\mathsf{T}$, derived in (21), is

$$AQD_{\text{ck}}(\mathbf{x}, \mathbf{b}) \;=\; \sum_{i=1}^{m} \big\|\mathbf{z}^{(i)} - D^{(i)}\mathbf{b}^{(i)}\big\|_2^2 + \|{R^\perp}^\mathsf{T}\mathbf{x}\|_2^2\,. \tag{25}$$

Here, $\big\|\mathbf{z}^{(i)} - D^{(i)}\mathbf{b}^{(i)}\big\|_2^2$ is the distance between the i^{th} projection of $\mathbf{x}$, *i.e.,* $\mathbf{z}^{(i)}$, and a subcenter projection from $D^{(i)}$ selected by $\mathbf{b}^{(i)}$. Given a query, these distances for each i, and all h possible values of $\mathbf{b}^{(i)}$ can be pre-computed and stored in a query-specific $h{\times}m$ lookup table. Once created, the lookup table is used to compare all database points to the query. So computing AQD_{ck} entails m lookups and additions, somewhat more costly than Hamming distance. The last term on the RHS of (25) is irrelevant for NN search. Since PQ is a special case of ck-means with pre-defined subspaces, the same distances are used for PQ (*cf.* [7]).

The SQD_{ck} between binary codes $\mathbf{b}_1$ and $\mathbf{b}_2$ is given by

$$SQD_{\text{ck}}(\mathbf{b}_1, \mathbf{b}_2) \;=\; \sum_{i=1}^{m} \big\|D^{(i)}\mathbf{b}_1^{(i)} - D^{(i)}\mathbf{b}_2^{(i)}\big\|_2^2\,. \tag{26}$$

Since $\mathbf{b}_1^{(i)}$ and $\mathbf{b}_2^{(i)}$ are 1-of-h encodings, an $m{\times}h{\times}h$ lookup table can be created to store all pairwise sub-distances.

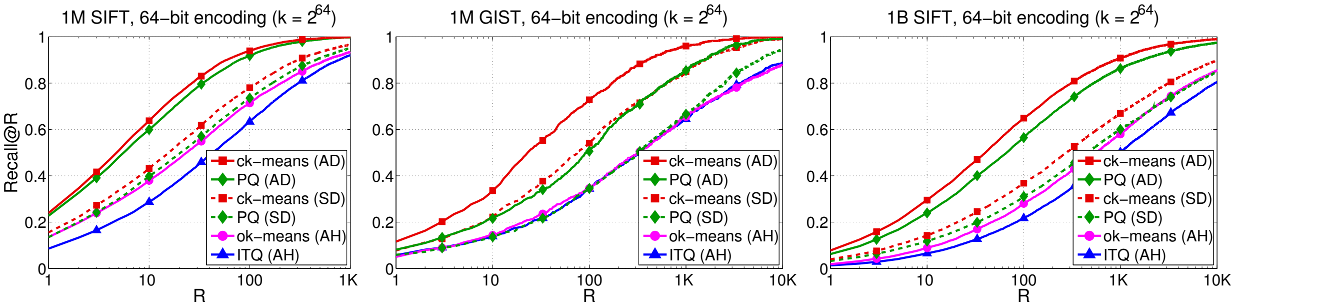

Figure 4. Euclidean NN recall@R (number of items retrieved) based on different quantizers and corresponding distance functions on the *1M SIFT*, *1M GIST*, and *1B SIFT* datasets. The dashed curves use symmetric distance. ($AH \equiv AQD_{\text{ok}}, SD \equiv SQD_{\text{ck}}, AD \equiv AQD_{\text{ck}}$)

While the cost of computing SQD_{ck} is the same as AQD_{ck}, SQD_{ck} could also be used to estimate the distance between the indexed database entries, for diversifying the retrieval results, or to detect near duplicates, for example.

Datasets. We use the *1M SIFT* dataset, as in Sec. 3.3, along with two others, namely, *1M GIST* (960D features) and *1B SIFT*, both comprising disjoint sets of training, base and test vectors. *1M GIST* has 500K training, 1M base, and 1K query vectors. *1B SIFT* has 100M training, 1B base, and 10K query points. In each case, recall rates are averaged over queries in test set for a database populated from the base set. For expedience, we use only the first 1M training points for the *1B SIFT* experiments.

Parameters. In ANN experiments below, for both ck-means and PQ, we use $m = 8$ and $h = 256$. Hence the number of clusters is $k = 256^8 = 2^{64}$, so 64-bits are used as database indices. Using $h\!=\!256$ is particularly attractive because the resulting lookup tables are small, encoding is fast, and each subcenter index fits into one byte. As h increases we expect retrieval results to improve, but encoding and indexing of a query to become slower.

Initialization. To initialize the D^i's for learning, as in k-means, we simply begin with random samples from the set of Z^i's (see Sec. 4.1). To initialize R we consider the different methods that Jégou et al. [7] proposed for splitting the feature dimensions into m sub-vectors for PQ: (1) *natural*: sub-vectors comprise consecutive dimensions, (2) *structured*: dimensions with the same index modulo 8 are grouped, and (3) *random*: random permutations are used.

For PQ in the experiments below, we use the orderings that produced the best results in [7], namely, the structured ordering for 960D GIST, and the natural ordering for 128D SIFT. For learning ck-means, R is initialized to the identity with SIFT corpora. For *1M GIST*, where the PQ ordering is significant, we consider all three orderings to initialize R.

Results. Fig. 4 shows Recall@R plots for ck-means and PQ [7] with symmetric and asymmetric distances ($SD \equiv SQD_{\text{ck}}$ and $AD \equiv AQD_{\text{ck}}$) on the 3 datasets. The horizontal axis represents the number of retrieved items, R, on a log-scale. The results consistently favor ck-means. On the high-dimensional GIST data, ck-means with AD significantly outperforms other methods; even ck-means with SD performs on par with PQ with AD. On *1M SIFT*, the Recall@10 numbers for PQ and ck-means, both using AD, are 59.9% and 63.7%. On *1B SIFT*, Recall@100 numbers are 56.5% and 64.9%. As expected, with increasing dataset size, the difference between methods is more significant.

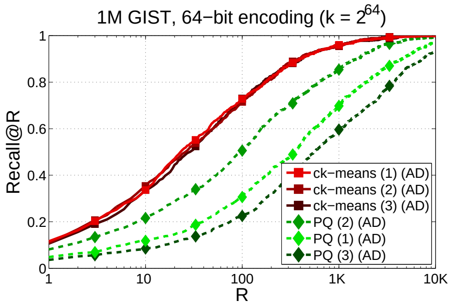

Figure 5. PQ and ck-means results using natural (1), structured (2), and random (3) ordering to define the (initial) subspaces.

In *1B SIFT*, each feature vector is 128 bytes, hence a total of 119 GB. Using any method in Fig. 4 (including ck-means) to index the database into 64 bits, this storage cost reduces to only 7.5 GB. This allows one to work with much larger datasets. In the experiments we use linear scan to find the nearest items according to quantizer distances. For NN search using 10K SIFT queries on *1B SIFT* this takes about 8 hours for AD and AH and 4 hours for Hamming distance on a 2×4-core computer. Search can be sped up significantly; using a coarse initial quantization and an inverted file structure for AD and AH, as suggested by [7, 1], and using the multi-index hashing method of [13] for Hamming distance. In this paper we did not implement these efficiencies as we focus primarily on the quality of quantization.

Fig. 5 compares ck-means to PQ when R in ck-means is initialized using the 3 orderings of [7]. It shows that ck-means is superior in all cases. Simiarly interesting, it also shows that despite the non-convexity of the optimization objective, ck-means learning tends to find similarly good encodings under different initial conditions. Finally, Fig. 6 compares ckmeans to PQ with different numbers of centers.

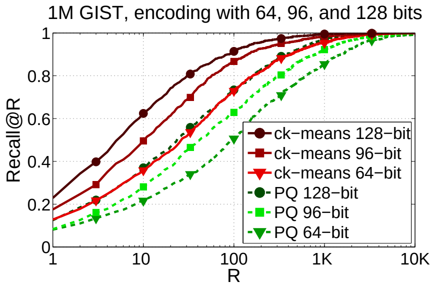

Figure 6. PQ and ck-means results using different number of bits for encoding. In all cases asymmetric distance is used.

Codebook	Accuracy
PQ ($k = 40^2$)	75.9%
ck-means ($k = 40^2$)	78.2%
k-means ($k = 1600$) [2]	77.9%
PQ ($k = 64^2$)	78.2%
ck-means ($k = 64^2$)	79.7%
k-means ($k = 4000$) [2]	79.6%

Table 2. Recognition accuracy on the CIFAR-10 test set using different codebook learning algorithms.

6.2. Learning visual codebooks

While the ANN seach tasks above require too many clusters for k-means, it is interesing to compare k-means and ck-means on a task with a moderate number of clusters. To this end we consider codebook learning for bag-of-words models [3, 10]. We use ck-means with $m = 2$ and $h = \sqrt{k}$, and hence k centers. The main advantage of ck-means here is that finding the closest cluster center is done in $O(\sqrt{k})$ time, much faster than standard NN search with k-means in $O(k)$. Alternatives for k-means, to improve efficiency, include approximate k-means [14], and hierarchical k-means [12]. Here we only compare to exact k-means.

CIFAR-10 [9] comprises 50K training and 10K test images (32×32 pixels). Each image is one of 10 classes (airplane, bird, car, cat, deer, dog, frog, horse, ship, and truck). We use publicly available code from Coates et al [2], with changes to the codebook learning and cluster assignment modules. Codebooks are built on 6×6 whitened color image patches. One histogram is created per image quadrant, and a linear SVM is applied to $4k$-dim feature vectors.

Recognition accuracy rates on the test set for different models and k are given in Table 2. Despite having fewer parameters, ck-means performs on par or better than k-means. This is consistent for different initializations of the algorithms. Although k-means has higher fedility than ck-means, with fewer parameters, ck-means may be less susceptible to overfitting. Table 2, also compares with the approach of [19], where PQ without learning a rotation is used for clustering features. As expected, learning the rotation has a significant impact on recognition rates, outperforming all three initializations of PQ.

7. Conclusions

We present the Cartesian k-means algorithm, a generalization of k-means with a parameterization of the cluster centers such that number of centers is super-linear in the number of parameters. The method is also shown to be a generalization of the ITQ algorithm and Product Quantization. In experiments on large-scale retrieval and codebook learning for recognition the results are impressive, outperforming product quantization by a significant margin. An implementation of the method is available at `https://github.com/norouzi/ckmeans`.

References

[1] A. Babenko and V. Lempitsky. The inverted multi-index. *CVPR*, 2012.

[2] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. *AISTATS*, 2011.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *ECCV Workshop Statistical Learning in Computer Vision*, 2004.

[4] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. *CVPR*, 2013.

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

[6] A. Gordo and F. Perronnin. Asymmetric distances for binary embeddings. *CVPR*, 2011.

[7] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 2011.

[8] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. *CVPR*, 2010.

[9] A. Krizhevsky. Learning multiple layers of features from tiny images. *MSc Thesis, Univ. Toronto*, 2009.

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006.

[11] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. IT*, 28(2):129–137, 1982.

[12] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *CVPR*, 2006.

[13] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. *CVPR*, 2012.

[14] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR*, 2007.

[15] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31, 1966.

[16] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. *ICCV*, 2003.

[17] J. Tenenbaum and W. Freeman. Separating style and content with bilinear models. *Neural Comp.*, 12:1247–1283, 2000.

[18] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE T. PAMI*, 29(5):854–869, 2007.

[19] S. Wei, X. Wu, and D. Xu. Partitioned k-means clustering for fast construction of unbiased visual vocabulary. *The Era of Interactive Media*, 2012.