The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Product Quantized Translation for Fast Nearest Neighbor Search

**Yoonho Hwang, Mooyeol Baek, Saehoon Kim,*
Bohyung Han, Hee-Kap Ahn**
Dept. of Computer Science and Engineering
POSTECH, Korea
{cypher, mooyeol, kshkawa, bhhan, heekap}@postech.ac.kr

Abstract

This paper proposes a simple nearest neighbor search algorithm, which provides the exact solution in terms of the Euclidean distance efficiently. Especially, we present an interesting approach to improve the speed of nearest neighbor search by proper translations of data and query although the task is inherently invariant to the Euclidean transformations. The proposed algorithm aims to eliminate nearest neighbor candidates effectively using their distance lower bounds in nonlinear embedded spaces, and further improves the lower bounds by transforming data and query through product quantized translations. Although our framework is composed of simple operations only, it achieves the state-of-the-art performance compared to existing nearest neighbor search techniques, which is illustrated quantitatively using various large-scale benchmark datasets in different sizes and dimensions.

Introduction

Nearest neighbor search is a fundamental technique applicable to various problems in machine learning, computer vision, natural language processing, information retrieval, and so on. They have been investigated actively in various fields of study, and a lot of promising results have been reported. However, it is not straightforward to identify the nearest neighbor efficiently from large-scale databases in very high dimensional spaces due to huge computational cost and inherent characteristics of data distribution (Beyer et al. 1999).

The most naïve approach for this problem is sequential scan, which simply computes the distance from a query to every entry in database iteratively. Although the running time of this technique is proportional to database size and dimensionality, it often shows the state-of-the-art performance in very high dimensional data, because the distances to the nearest neighbor and the farthest neighbor are not differentiated sufficiently and the benefit of algorithms based on sophisticated data structures tends to become negligible compared to their costs (Weber, Schek, and Blott 1998).

Many existing techniques for the exact nearest neighbor search employ tree-based data structures and manage data through hierarchical partitioning. Representative examples include KD-tree (Bentley 1975; Arya et al. 1998), and Cover tree (Beygelzimer, Kakade, and Langford 2006). They typically aim to improve search performance in moderately high dimensional space, and do not suit well for high-dimensional data due to their equidistant property. In addition, these methods require significant amount of time and memory space for data structure construction and management (Karger and Ruhl 2002). Another type of exact algorithms incorporates data embedding tricks, where original data are projected onto low dimensional subspaces either linearly (Hel-Or and Hel-Or 2005) or nonlinearly (Hwang, Han, and Ahn 2012). However, it turns out that the linear projection method based on Walsh-Hadamard transform (Hel-Or and Hel-Or 2005) is identical to the sequential scan if it is applied to the data without inter-dependency[1]. There also exist various techniques to improve throughput of the algorithm by parallel implementation or batch run of multiple queries, but they are beyond the scope of this paper.

Instead of computing exact solutions, some algorithms find approximate nearest neighbors in which they focus on reducing computational complexity while minimizing loss of accuracy. Approximate Nearest Neighbor (ANN) search algorithm (Arya et al. 1998) employs a KD-tree to maintain data, and other techniques based on KD-trees have also been presented (Silpa-Anan and Hartley 2008; Jia et al. 2010). In general, they achieve fairly good performance but require substantial preprocessing time and memory. FLANN (Muja and Lowe 2014) constructs multiple tree-based data structures including randomized KD-trees, priority search k-means tree and hierarchical clustering tree, and attempts to improve approximate nearest neighbor search speed by integrating a few additional techniques. Product Quantization (PQ) approaches generate a large number of data clusters through the combinations of clusters in multiple axis-aligned subspaces and compute the distances to individual data from a query efficiently using the precomputed distances between identified cluster centers in the subspaces. It is first introduced in (Jégou, Douze, and Schmid 2011) and an optimized solution is discussed in (Ge et al. 2013). Hashing algorithms typically manage a hash table and reduce nearest neighbor candidates significantly using

*S.Kim is also affiliated with AItrics.

[1]An example showing the benefit of (Hel-Or and Hel-Or 2005) is dense nearest neighbor search between all pairs of patches from images.

an associated hash function. Examples in this category include locality-sensitive hashing (Indyk and Motwani 1998; Datar et al. 2004), spectral hashing (Weiss, Torralba, and Fergus 2008), and coherency sensitive hashing (Korman and Avidan 2011). Although hashing is effective to find approximate nearest neighbors, it is not appropriate for exact search due to rapid increase of computational cost to resolve collisions.

Our goal is to find the *exact* nearest neighbor efficiently from large-scale databases in terms of the Euclidean distance with no constraint on data distribution. The proposed algorithm belongs to the same category with (Hwang, Han, and Ahn 2012) while it has a few interesting new features for performance enhancement. The main idea of (Hwang, Han, and Ahn 2012) is to reject non-nearest neighbors by efficiently computing distance lower bounds from a query, where the lower bound of distance between each data point and the query is given by their element means and variances.

We propose a product quantized translation algorithm to increase the distance lower bounds for filtering more examples and improve empirical time complexity of the nearest neighbor search. Our algorithm is combined with a filtering-based nearest neighbor search technique (Hwang, Han, and Ahn 2012), and achieves the state-of-the-art performance in various datasets. For the purpose, we precompute a quantized translation of each data point in database and apply the same translation to a query to identify the nearest neighbor by filtering. To determine a proper quantized translation vector for each example, we perform clustering all data points in database and use the membership information of each point to the identified clusters. This idea sounds a bit weird since nearest neighbor search is invariant to translation, but we will show how to improve the distance lower bound statistically by translating data.

The remaining issue is that the number of clusters required to improve the distance lower bound through data translations increases rapidly as their dimensionality grows, which makes query time significantly slower. Hence, we adopt the idea from product quantization (Jégou, Douze, and Schmid 2011) to solve this issue, where the clusters in the full dimension are obtained from a Cartesian product of subspace clusters. We need to maintain few clusters in each subspace while an extremely large number of clusters are generated by the combinations of subspace clusters.

The contributions and characteristics of this paper are summarized below:

- We observe that the distance lower bounds (Hwang, Han, and Ahn 2012) of points in a group from a query are improved by a proper translation (same translation for all the points in the group).
- To achieve better lower bounds, we employ product quantization (Jégou, Douze, and Schmid 2011; Ge et al. 2013) and generate a extremely large number of quantized translation vectors.
- The proposed algorithm illustrates outstanding performance in four large-scale datasets without any sophisticated data structures and large memory requirement.

The rest of the paper is organized as follows. We first summarize two important prior works closely related to our algorithm. Then, the main idea of our algorithm and its technical details are presented. In the section for experiment, the proposed algorithm is evaluated in four public benchmark datasets and its effectiveness is illustrated with respect to other exact nearest neighbor search methods.

Background

This section reviews two prior works closely related to our approach, nonlinear embedding for exact nearest neighbor search (Hwang, Han, and Ahn 2012) and product quantization for approximate nearest neighbor search (Jégou, Douze, and Schmid 2011).

Nearest Neighbor Search by Nonlinear Embedding

An interesting method for efficient nearest neighbor search has been proposed in (Hwang, Han, and Ahn 2012), which embeds data points onto a low-dimensional space and filters non-nearest neighbors out using the distances in the embedded space.

Denote a data point in database $\mathcal{X}$ by $\mathbf{x} = (x_1, \ldots, x_d)^\mathsf{T}$. The mean and variance of the elements in $\mathbf{x} \in \mathbb{R}^d$ are

$$\mu_\mathbf{x} = \frac{1}{d}\sum_{i=1}^{d} x_i \text{ and } \sigma_\mathbf{x}^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu_\mathbf{x})^2, \qquad (1)$$

respectively. If a query $\mathbf{y} = (y_1, \ldots, y_d)^\mathsf{T}$ is presented, a lower bound on the squared Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$, $\|\mathbf{x} - \mathbf{y}\|^2$, is given by

$$\|\mathbf{x}-\mathbf{y}\|^2 \geq d\left((\mu_\mathbf{x}-\mu_\mathbf{y})^2+(\sigma_\mathbf{x}-\sigma_\mathbf{y})^2\right) \equiv \text{LB}(\mathbf{x},\mathbf{y}). \qquad (2)$$

Since $\mu_\mathbf{x}$ and $\sigma_\mathbf{x}$ can be precomputed, we need to compute $\mu_\mathbf{y}$ and $\sigma_\mathbf{y}$ once in a query time and $\text{LB}(\mathbf{x}, \mathbf{y})$ for any $\mathbf{x} \in \mathcal{X}$ is computed in $O(1)$. Then, the nearest neighbor search from a query is performed efficiently by filtering candidates based on the lower bound.

Product Quantization

Product quantization (Jégou, Douze, and Schmid 2011; Ge et al. 2013) is a technique for approximate nearest neighbor search, which quantizes data in multiple exclusive low-dimensional axis-aligned subspaces and computes approximate distances using the quantized vectors obtained from a Cartesian product of the subspaces. The main advantage of product quantization is that a huge number of quantized vectors in the original dimension are generated using only a few quantized codewords in each subspace through Cartesian product. Note that it is not necessary to store the huge number of quantized vectors in full dimension while we need to store only a few codewords in each subspace and their pairwise distances.

A data $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ is divided into m subvectors, which is denoted by $\mathbf{u}_j \in \mathcal{X}_j$ $(j = 1, \ldots, m)$. Each subvector $\mathbf{u}_j$ is quantized separately by either clustering or encoding function. Formally, $\mathbf{x} = (\mathbf{u}_1^\mathsf{T}, \ldots, \mathbf{u}_m^\mathsf{T})^\mathsf{T}$ is quantized to $\hat{\mathbf{q}}(\mathbf{x}) = (\mathbf{q}_1(\mathbf{u}_1)^\mathsf{T}, \ldots, \mathbf{q}_m(\mathbf{u}_m)^\mathsf{T})^\mathsf{T}$, where $\mathbf{q}_j : \mathcal{X}_j \mapsto \mathcal{C}_j$ is a quantization function on each of m partitioned subspaces.

The distance from a query $\mathbf{y} = (\mathbf{v}_1^\top, \ldots, \mathbf{v}_m^\top)^\top$ to $\mathbf{x}$ is computed using the codewords approximately as

$$\|\mathbf{x}-\mathbf{y}\| \approx \left(\sum_{j=1}^{m} \|\mathbf{q}_j(\mathbf{u}_j) - \mathbf{v}_j\|^2\right)^{1/2}. \tag{3}$$

Since points in database are quantized, it is possible to compute all the distances from a query approximately by computing distances with the codewords. Moreover, due to the benefit of product quantization, we maintain small quantization errors by generating a huge number of quantized codes using a small number of clusters in each subspace.

Main Algorithm

Our algorithm computes the nearest neighbor by filtering data based on their distance lower bounds given by a nonlinear embedding as in (Hwang, Han, and Ahn 2012). For more aggressive filtering, we identify translation vectors to improve distance lower bounds between two points. This section describes why this technique is useful to improve the lower bounds and requires an extremely large number of translation vectors for better performance. We also discuss how to generate sufficiently many translation vectors by product quantization technique.

Translation for Exact Nearest Neighbor Search

The nearest neighbor search is invariant to translation, *i.e.*, $\|\mathbf{x}-\mathbf{y}\| = \|(\mathbf{x}-\mathbf{t}) - (\mathbf{y}-\mathbf{t})\|, \forall \mathbf{t} \in \mathbb{R}^d$ while the distance lower bound defined in Eq. (2) is affected by the translation as shown in

$$\begin{aligned}\mathrm{LB}(\mathbf{x}-\mathbf{t}, \mathbf{y}-\mathbf{t}) &= d\left((\mu_{\mathbf{x-t}} - \mu_{\mathbf{y-t}})^2 + (\sigma_{\mathbf{x-t}} - \sigma_{\mathbf{y-t}})^2\right) \\ &= d\left((\mu_{\mathbf{x}} - \mu_{\mathbf{y}})^2 + (\sigma_{\mathbf{x-t}} - \sigma_{\mathbf{y-t}})^2\right). \end{aligned} \tag{4}$$

If $\mathbf{t}$ gets close to $\mathbf{x}$, *i.e.*, $\mathbf{t} \approx \mathbf{x}$, the following equation holds:

$$\begin{aligned}\mathrm{LB}(\mathbf{x}-\mathbf{t}, \mathbf{y}-\mathbf{t}) &\approx d(\mu_{\mathbf{x-y}}^2 + \sigma_{\mathbf{x-y}}^2) \\ &= \|\mathbf{x}-\mathbf{y}\|^2, \end{aligned} \tag{5}$$

which means that the lower bound is optimized when $\mathbf{t} = \mathbf{x}$. When we find the nearest neighbor among $\mathbf{x}_i \in \mathcal{X}$, one option is to translate query $\mathbf{y}$ by every $\mathbf{x}_i$ and compute the distance between $\mathbf{y}$ and $\mathbf{x}_i$, which is time-consuming because it becomes equivalent to sequential scan. Instead of translating $\mathbf{y}$ by all the vectors in $\mathcal{X}$, we aim to reduce the number of translations while maintaining tight lower bounds. This objective is achieved by quantized translations, for which we should identify good translation vectors.

To maximize the benefit of quantized translation, we need to identify tightly coupled clusters and apply the same translation to all the points in a cluster. The objective function corresponding to this goal is given by

$$\underset{\{\mathcal{X}^l, \mathbf{t}^l\}_{l=1:k}}{\operatorname{argmin}} \sum_{l} \sum_{\mathbf{x} \in \mathcal{X}^l} ||\mathbf{x} - \mathbf{t}^l||^2, \tag{6}$$

which is equivalent to the objective of k-means clustering. In other words, by integrating k-means clustering algorithm, we learn a set of proper translation vectors and achieve the improved distance lower bound; the identified k centroids and cluster membership information are used for translation. Figure 1 illustrates the procedure to generate a set of proper translations and improve the distance lower bounds for the 2D examples.

On the other hand, it is unlikely to reduce quantization error substantially only with a small number of translation candidates. In the next subsection, we present a technique that generates an extremely large number of translations effectively and selects the best one efficiently through combinations of subspace translations.

Product Quantized Translation for Nearest Neighbor Search

When the dimensionality is very high and there are a large number of data, it is unlikely that one can reduce quantization error using a small number of candidate translations effectively. To resolve this issue, we introduce product quantized translation (PQT) that borrows some idea from product quantization. In PQT, a few quantized vectors are obtained from each subspace independently and the full quantizations are generated by a Cartesian product of subspace quantizations (Jégou, Douze, and Schmid 2011; Ge et al. 2013).

If a data point $\mathbf{x} \in \mathcal{X}$ is divided into m subspaces as $\mathbf{x} = (\mathbf{u}_1^\top, \ldots \mathbf{u}_m^\top)^\top$, we precompute a quantized translation $\mathbf{q}_j(\mathbf{u}_j) \in \mathcal{C}_j$ $(j = 1, \ldots, m)$ for each $\mathbf{u}_j \in \mathbb{R}^{d_j}$ and generate a large number of translation candidates in the full dimension $\hat{\mathbf{q}}(\mathbf{x}) = (\mathbf{q}_1(\mathbf{u}_1)^\top, \ldots, \mathbf{q}_m(\mathbf{u}_m)^\top)^\top$.

We optimize PQT by adopting the alternative-optimization procedure introduced in (Ge et al. 2013), which finds a good space decomposition matrix for dividing subspaces and the clusters in each subspace. After optimization, $\mu_{\mathbf{u}_j}$ and $\sigma_{\mathbf{u}_j - \mathbf{q}(\mathbf{u}_j)}$ for $\forall \mathbf{x} \in \mathcal{X}$ are also computed in preprocessing step. When a query $\mathbf{y} = (\mathbf{v}_1^\top, \ldots, \mathbf{v}_m^\top)^\top$ is given, $\mu_{\mathbf{v}_j}$ and $\sigma_{\mathbf{v}_j - \mathbf{c}_j}$ for all codeword $\mathbf{c}_j \in \mathcal{C}_j$ $(j = 1, \ldots, m)$ are computed once. Then for any translation vector $\mathbf{q}_j(\mathbf{u}_j) \in \mathcal{C}_j$, its corresponding $\sigma_{\mathbf{v}_j - \mathbf{q}_j(\mathbf{u}_j)}$ can be achieved directly. This procedure is symmetric to product quantization as shown in Table 1.

Progressive Filtering by PQT

On query time, our algorithm efficiently filters out non-nearest neighbors using the distance lower bounds discussed above. We compute the distance lower bound by augmenting the lower bounds in subspaces progressively. Let $\hat{\mathrm{LB}}_j(\mathbf{x}, \mathbf{y})$ be the cumulative lower bound of the distance between $\mathbf{x}$ and $\mathbf{y}$ in the j^{th} subspace, which is given by

$$\hat{\mathrm{LB}}_0(\mathbf{x}, \mathbf{y}) \equiv 0 \tag{7}$$

$$\hat{\mathrm{LB}}_j(\mathbf{x}, \mathbf{y}) \equiv \hat{\mathrm{LB}}_{j-1}(\mathbf{x}, \mathbf{y}) + \mathrm{LB}(\mathbf{u}_j - \mathbf{q}_j(\mathbf{u}_j), \mathbf{v}_j - \mathbf{q}_j(\mathbf{u}_j)). \tag{8}$$

If $\hat{\mathrm{LB}}_j(\mathbf{x}, \mathbf{y})$ is larger than the distance to the current nearest neighbor, we reject $\mathbf{x}$. Otherwise, we repeat the same procedure to obtain a better lower bound with the next subspace.

It is still possible to fail to determine whether $\mathbf{x}$ is the nearest so far even after computing $\hat{\mathrm{LB}}_m(\mathbf{x}, \mathbf{y})$. In this case, our algorithm comes back to the first partition and update its

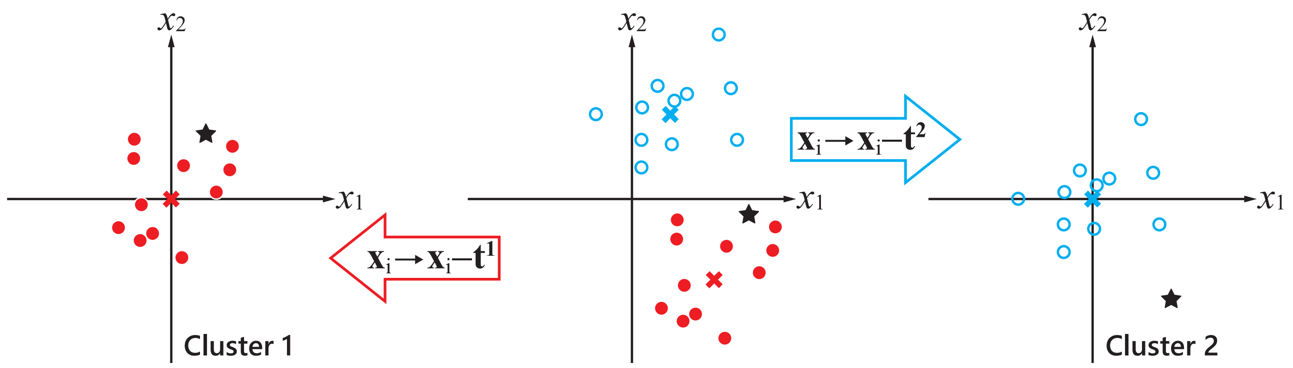

x_2
x_1
Cluster 1
$\mathbf{x}_i \rightarrow \mathbf{x}_i - \mathbf{t}^1$
$\mathbf{x}_i \rightarrow \mathbf{x}_i - \mathbf{t}^2$
Cluster 2

Figure 1: Procedure to identify translations through 2-means clustering for 2-dimensional data. Two cluster centers are denoted by red and blue x marks. Note that, if we apply the translation to each cluster ($\mathbf{t}^1$ for cluster 1 and $\mathbf{t}^2$ for cluster 2) so that the cluster center becomes the origin, there are more possibilities to improve the distance lower bounds and to reject more examples statistically given a query denoted by black star with the same translation.

Table 1: Symmetry between product quantization and PQT

	Product Quantization	PQT
Target to quantize	data vectors $\mathbf{x} \in \mathcal{X}$	translations for $\mathbf{x} \in \mathcal{X}$
Optimization	$\sum \|\mathbf{x} - \hat{\mathbf{q}}(\mathbf{x})\|^2$, $\hat{\mathbf{q}} : \mathcal{X} \mapsto \mathcal{C}$	$\sum \|\mathbf{x} - \hat{\mathbf{q}}(\mathbf{x})\|^2$, $\hat{\mathbf{q}} : \mathcal{X} \mapsto \mathcal{C}$
Query time generation	$\|\mathbf{y} - \mathbf{c}\|^2$,$\forall \mathbf{c} \in \mathcal{C}$	$\sigma_{\mathbf{y}-\mathbf{c}}$,$\forall \mathbf{c} \in \mathcal{C}$

Algorithm 1 Nearest neighbor search by product quantized translation

```
 1: Pick a vector in $\mathcal{X}$ as the seed $\mathbf{x}^{\min}$
 2: MINDIST $\leftarrow \|\mathbf{x}^{\min} - \mathbf{y}\|^2$
 3: Compute $\mu_{\mathbf{v}_j}$ and $\sigma_{\mathbf{v}_j - \mathbf{q}(\mathbf{v}_j)}$ $(j = 1, \ldots, m)$
 4: for $\mathbf{x} \in \mathcal{X}$ do
 5:     for $j = 1 \rightarrow m$ do
 6:         if $\hat{\mathrm{LB}}_j(\mathbf{x}, \mathbf{y}) \geq$ MINDIST then
 7:             continue    // the first stage filtering
 8:     for $j = 1 \rightarrow m$ do
 9:         if $\mathrm{LB}^+_j(\mathbf{x}, \mathbf{y}) \geq$ MINDIST then
10:             continue    // the second stage filtering
11:     if $\mathrm{LB}^+_m(\mathbf{x}, \mathbf{y}) <$ MINDIST then
12:         $\mathbf{x}^{\min} \leftarrow \mathbf{x}$
13:         MINDIST $\leftarrow \mathrm{LB}^+_m(\mathbf{x}, \mathbf{y})$
14: return $\{\mathbf{x}^{\min}$, MINDIST$\}$
```

distance lower bound with the exact distance to increase the lower bound, which is given by

$$\hat{\mathrm{LB}}^+_0(\mathbf{x}, \mathbf{y}) \equiv \hat{\mathrm{LB}}_m(\mathbf{x}, \mathbf{y}) \tag{9}$$

$$\hat{\mathrm{LB}}^+_j(\mathbf{x}, \mathbf{y}) \equiv \hat{\mathrm{LB}}^+_{j-1}(\mathbf{x}, \mathbf{y}) - \mathrm{LB}(\mathbf{u}_j - \mathbf{q}_j(\mathbf{u}_j), \mathbf{v}_j - \mathbf{q}_j(\mathbf{u}_j)) + \|\mathbf{u}_j - \mathbf{v}_j\|^2 . \tag{10}$$

By increasing the distance lower bound as described, we can identify the exact nearest neighbor at some point.

Summary of Algorithm

First, we partition the input dimensions into multiple axis-aligned subspaces and augment the distance lower bound incrementally using the subspaces for data filtering. In this step, the idea for the product of quantized translations is employed to improve the lower bound. Second, if there still remain unfiltered examples, we convert the lower bound of each partition to the exact distance one by one and increase the lower bound. In the middle of the procedure, we stop at any time and identify the nearest neighbor when there is a single example unfiltered among all data in the database. The pseudocode of our algorithm is presented in Algorithm 1.

Since the algorithm uses the linear scan scheme, the search time in the best case is $O(n)$ in theory. In contrast, the worst case search time is $O(dn)$ when every filtering is failed. However, such a bad case hardly happens in practice, since we use extremely large number of translations with progressive filtering to increase the distance lower bounds. In the next section, we illustrate outperforming performance of our algorithm with detailed analysis on the improvement of lower bounds and filtering performance via quantized translations.

Experiment

The proposed algorithm is evaluated in four benchmark datasets with different characteristics, and the detailed results are presented in this section.

Datasets

We perform the experiments on four independent datasets, which are denoted by MNIST (Lecun et al. 1998), SIFT5M (Jégou, Douze, and Schmid 2011), GIST1M (Jégou, Douze, and Schmid 2011), and MS-COCO (Lin et al. 2014). MNIST is well-known machine

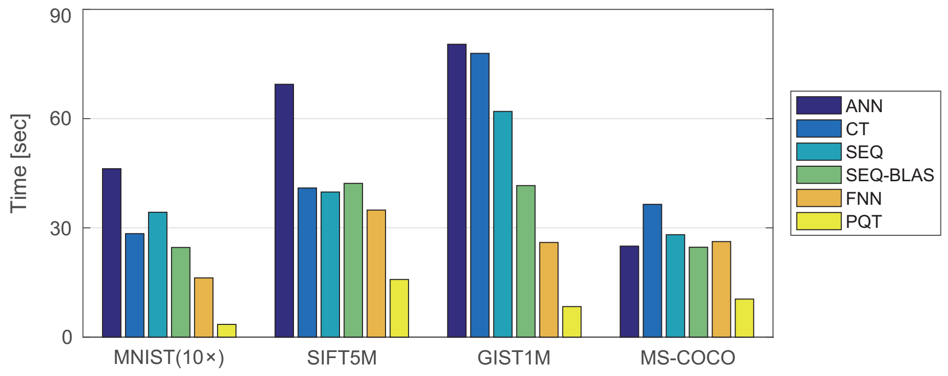


Figure 2: Query times of the compared algorithms in the four tested datasets. MNIST query times are scaled up by 10 times for readability. PQT is consistently better than all other methods. For PQT, the number of clusters and the dimensionality of partitions are set to 64 and 32, respectively.

learning dataset which consists of 70,000 784-dimensional handwritten digit images. SIFT5M has five million entries of 128-dimensional SIFT (Lowe 2004) feature descriptors while GIST1M is composed of one million 960-dimensional GIST (Oliva and Torralba 2001) feature descriptors. MS-COCO contains 4,096-dimensional vectors, which are feature descriptors for 123,287 images in training and validation sets from the last convolutional layer (fc7) in VGG-16 net (Simonyan and Zisserman 2015). Individual datasets have their unique characteristics; data dimensions are diverse, from 128 to 4096, and the numbers of data are from as small as 70 thousand and to as large as 5 million. In addition, data in MNIST, SIFT5M, and GIST1M are typically located in low- or moderate-dimensional manifolds while MS-COCO has substantial randomness and contains many outliers.

Implementation Details

All tested algorithms are implemented in C++ in Linux (Fedora 21, g++ 4.9.2), specifically using a single core on Intel Core i7-5820k@3.30Ghz with 64GB main memory. We use the source codes released by authors for the implementations of the external algorithms, and the results from all algorithms are reproduced with the default parameters in the original implementations.

Since our algorithm adopts the idea from the product quantization algorithm (Jégou, Douze, and Schmid 2011), we can naturally incorporate the existing optimization technique to alternate translations of the cluster centers and rotations of the original data (Ge et al. 2013) for preprocessing. Since k-means clustering and data rotations in the preprocessing step are computationally expensive, we employ 1% of examples with minimum 10,000 samples for the clustering.[2] We randomly select five seeds for initialization of our algorithm (for line 1 in Algorithm 1), which is similar to (Hwang, Han, and Ahn 2012). In all experiments, randomly selected 100 queries are executed one by one. By default, the number of clusters and the dimensionality of partitions are set to 64 and 32, respectively.

[2]When clustering and rotation are performed using 10% of data, the performance improvement is marginal.

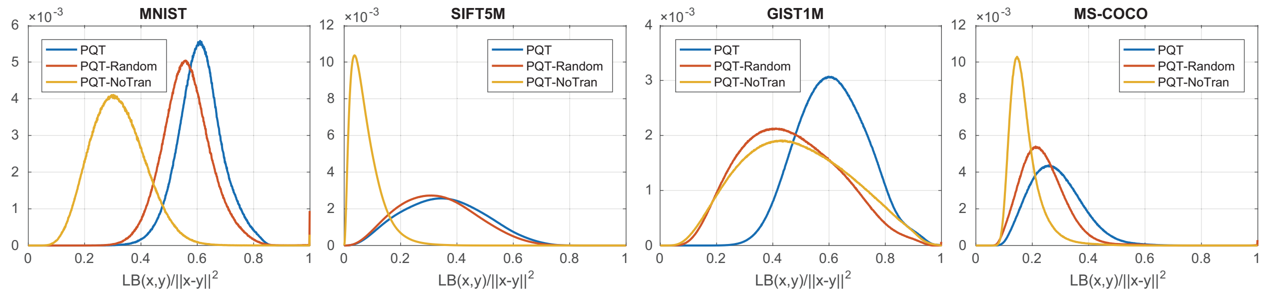
Results

We now present comprehensive results from our experiments in all tested datasets. Our algorithm is compared with a few important external algorithms that are able to report the exact nearest neighbor, which include: 1) optimized sequential scan (SEQ) that conditionally branches, 2) full search using OpenBLAS (Xianyi, Qian, and Yunquan 2012) library and precomputed norms (SEQ-BLAS), 3) nonlinear embedding method (FNN) (Hwang, Han, and Ahn 2012), 4) the exact version of approximate nearest neighbor search with KD-tree (ANN) (Arya et al. 1998), and 5) cover tree (CT) (Beygelzimer, Kakade, and Langford 2006). The proposed algorithm is denoted by product quantized translation (PQT). We also present an analysis on the behavior of our algorithm by investigating how much the lower bounds are improved by our approach in each dataset.

Quantitative results Figure 2 summarizes the quantitative results of all compared algorithms in all datasets, where our algorithm presents the state-of-the-art performance consistently. Except our algorithm, another filtering-based approach, FNN, shows relatively good performance.

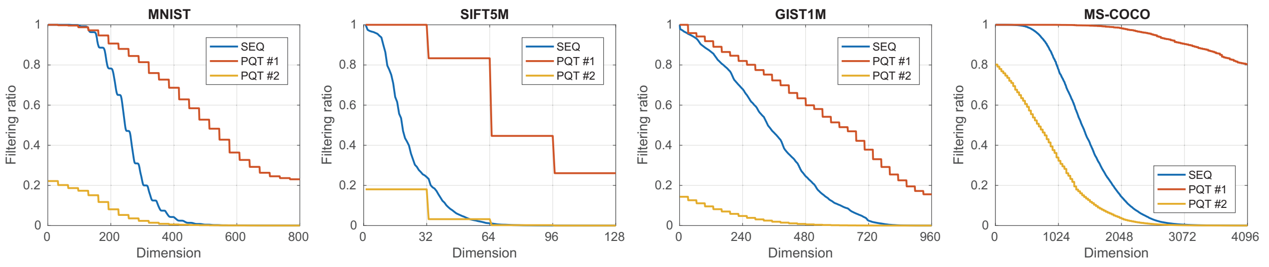
To understand how much the distance lower bounds are improved by identifying appropriate translations, we evaluate the quality of distance lower bound in our algorithm for three choices of quantized translations: PQT, PQT-Random, and PQT-NoTran. PQT is the algorithm with product quantized translations using k-means clustering, PQT-Random is based on clustering with k random centroids, and PQT-NoTran is without any translation. Figure 3 shows the improvement of lower bound quality due to the translations in PQT.

Filtering performance Figure 4 illustrates how many examples are filtered by our algorithm and SEQ in each dataset. The filtering ratio of PQT #1 and PQT #2 are measured at the first and second filtering stage in our Algo-



(a) MNIST (b) SIFT5M (c) GIST1M (d) MS-COCO

Figure 3: The probability density of LB quality $(\mathrm{LB}(\mathbf{x},\mathbf{y})/\left\|\mathbf{x}-\mathbf{y}\right\|^2)$ with different clustering options in the tested datasets. The quality is substantially improved with proper product quantized translations identified by k-means clustering.



(a) MNIST (b) SIFT5M (c) GIST1M (d) MS-COCO

Figure 4: Filtering performance of the proposed algorithm in the four tested datasets. Each graph shows the ratio of remaining examples with respect to the number of dimensions.

Table 2: Query times of our algorithm in seconds with several different parameter sets. We did not present the results of 64D cases in SIFT5M because its partition is too coarse compared to the full dimensions and the results are not meaningful.

		MNIST			SIFT5M			GIST1M			MS-COCO		
# of clusters		32	64	128	32	64	128	32	64	128	32	64	128
Partition dimension	16	0.56	0.55	0.56	12.90	12.28	**11.57**	9.96	10.14	10.62	10.78	12.67	18.32
	32	0.36	0.35	0.35	17.46	15.83	14.60	**7.94**	8.39	9.07	**10.42**	10.45	12.66
	64	0.35	**0.31**	0.31	–	–	–	9.54	10.45	11.15	10.66	11.30	11.45

rithm 1, respectively. We compute the distance lower bound of an example in the embedded space after translation and reject the example based on the lower bound if possible. It would be better if we can eliminate many candidates at the early stages, but the rejection rates depend on the characteristics of data. According to our observation, a significant portion of data is rejected by the first stage of our filtering algorithm in MNIST, SIFT5M, and GIST1M. However, in MS-COCO, filtering ratio is relatively low because it involves a lot of randomness and the lower bounds computed in the embedded spaces are not sufficiently tight, unfortunately. Since many nearest neighbor search algorithms have similar issues in this kind of datasets, sequential scan often shows the best performance. However, even with such challenges, our algorithm still presents the state-of-the-art performance in MS-COCO, which indirectly demonstrates low overhead of the first stage filtering. On the other hand, filtering performance curves of SEQ are located consistently between the ones for the first and the second stages of our algorithm, which is reasonable considering overall time complexity of both algorithms. The good performance in MS-COCO is a great advantage since the feature descriptors obtained from convolutional neural networks are very discriminative and simple classifiers relying on the nearest neighbor search get more attention in various visual recognition problems.

Parameter settings Table 2 presents how much the performance of the proposed algorithm is affected by parameter setting. Specifically, we evaluate the speed of our algorithm by varying the number of clusters and the dimensionality of partitioned vectors, where three different values are tested for each parameter. The bold-faced numbers in Table 2 indicate the best-performing combinations of the two parameters in the benchmark datasets. Although our algorithm has moderate variations in performance with respect to the choices of parameters, it illustrates reasonably good performance in all datasets regardless of parameter setting.

Conclusion

We proposed a novel exact nearest neighbor search algorithm by product quantized translations and presented its computational efficiency compared with existing methods. We empirically show that the distance lower bound of a point from a query is improved by a proper quantized translation, and the distance lower bound is further improved by learning translation vectors. The proposed algorithm is useful to enhance the performance in filtering-based nearest neighbor search algorithms. It is possible to identify the best translation vector among a large number of precomputed candidates by the combination of the translations in subspaces. The proposed algorithm was evaluated in various independent datasets, and illustrates the state-of-the-art performance consistently.

Acknowledgements

This research was supported in part by the MSIT (Ministry of Science and ICT), Korea, under the SW Starlab support program (IITP–2017–0–00905) supervised by the IITP (Institute for Information & communications Technology Promotion.) and the IITP grant [2017-0-01778, Development of Explainable Human-level Deep Machine Learning Inference Framework; 2017-0-01780, The Technology Development for Event Recognition/Relational Reasoning and Learning Knowledge based System for Video Understanding; 2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion].

References

Arya, S.; Mount, D. M.; Netanyahu, N. S.; Silverman, R.; and Wu, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM* 45(6):891–923.

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Communications of ACM* 18(9):509–517.

Beyer, K.; Goldstein, J.; Ramakrishnan, R.; and Shaft, U. 1999. When is "nearest neighbor" meaningful? In *ICDT*.

Beygelzimer, A.; Kakade, S.; and Langford, J. 2006. Cover trees for nearest neighbor. In *ICML*.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, 253–262.

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

Hel-Or, Y., and Hel-Or, H. 2005. Real-time pattern matching using projection kernels. *TPAMI* 27(9):1430–1445.

Hwang, Y.; Han, B.; and Ahn, H.-K. 2012. A fast nearest neighbor search algorithm by nonlinear embedding. In *CVPR*.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 604–613.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *TPAMI* 33(1):117–128.

Jia, Y.; Wang, J.; Zeng, G.; Zha, H.; and Hua, X.-S. 2010. Optimizing KD-trees for scalable visual descriptor indexing. In *CVPR*.

Karger, D. R., and Ruhl, M. 2002. Finding nearest neighbors in growth-restricted metrics. In *STOC*, 741–750.

Korman, S., and Avidan, S. 2011. Coherency sensitive hashing. In *ICCV*.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollár, P. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. *IJCV* 60(2):91–110.

Muja, M., and Lowe, D. G. 2014. Scalable nearest neighbor algorithms for high dimensional data. *TPAMI* 36(11):2227–2240.

Oliva, A., and Torralba, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV* 42(3):145—175.

Silpa-Anan, C., and Hartley, R. 2008. Optimised KD-trees for fast image descriptor matching. In *CVPR*.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Weber, R.; Schek, H.-J.; and Blott, S. 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, 194–205.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *NIPS*, 1753–1760.

Xianyi, Z.; Qian, W.; and Yunquan, Z. 2012. Model-driven level 3 blas performance optimization on loongson 3a processor. In *ICPADS*.