Nearest Neighbor Search

Brief Survey

Ryo Matsuda

June 2021

Caveat (setting the bar low)

- None of the projects I’ve worked on has nothing to do with this topic
- I just googled for fun => Do not expect too much

I. Introduction

II. Prerequisites

III. Main Topic

What's Nearest Neighbor Search?

- Given N D-dim vectors (dataset): $X := \left\{x_i \in \mathbb{R}^D\right\}_{i=1,..,N}$
- For Q queries:
  - Given a query vector: $q \in \mathbb{R}^D$
  - Find: k-argmin$_i$ dist(q, x_i)
    - dist: $\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_{\geq 0}$
    - e.g. Euclidean distance, inner product, cosine similarity (argmax in this case)
- Typically, $D = \mathcal{O}(100)$, N is large (thousands, millions, billions, …)

- Brute Force linear search:
  - Space Complexity: $\mathcal{O}(ND)$
  - Time Complexity per query: $\mathcal{O}(ND)$
- Approximate solutions (= less accuracy) will do for better time & space complexity
  - Tradeoffs

Application

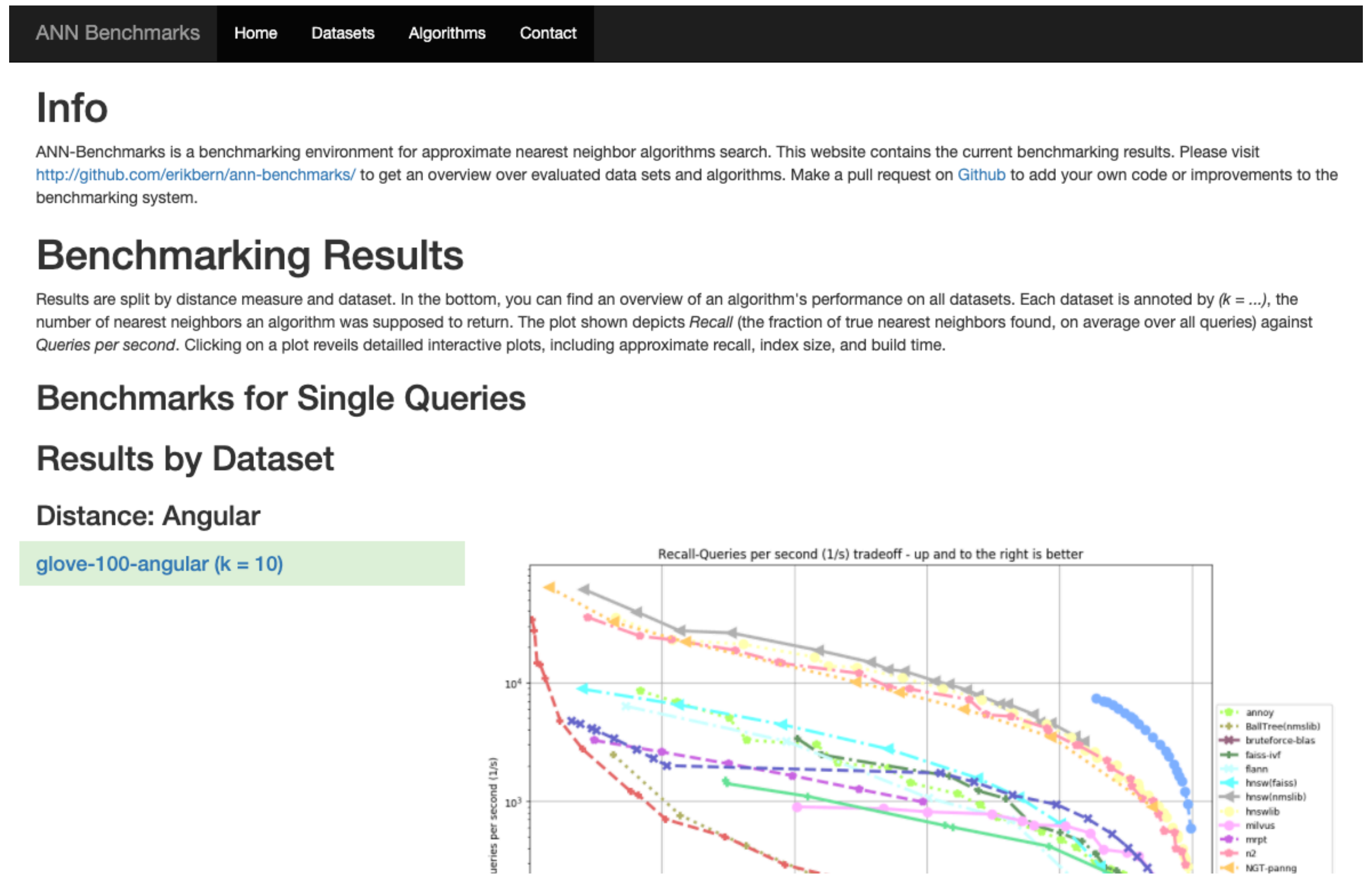
- Clustering
- Information Retrieval
  - Images
  - Texts
  - QA
    - REALM
- Recommendation System

Glossaries

- NN: Nearest Neighbor Search ~ Similarity Search
- MIPS: Maximum Inner Product Search
- ANN: Approximate Nearest Neighbor Search
- Exact Nearest Neighbor Search

It’s been studied a lot

- Algorithms
  - Hashing
  - Partitioning
  - Graph traversal
  - Compression/Quantization
- Tools
  - nmslib
  - FLANN
  - Falconn
  - Annoy
  - **Faiss**
  - **scaNN**



http://ann-benchmarks.com

References & Today's topic

<- AWESOME Tutorial
By Matsui-san

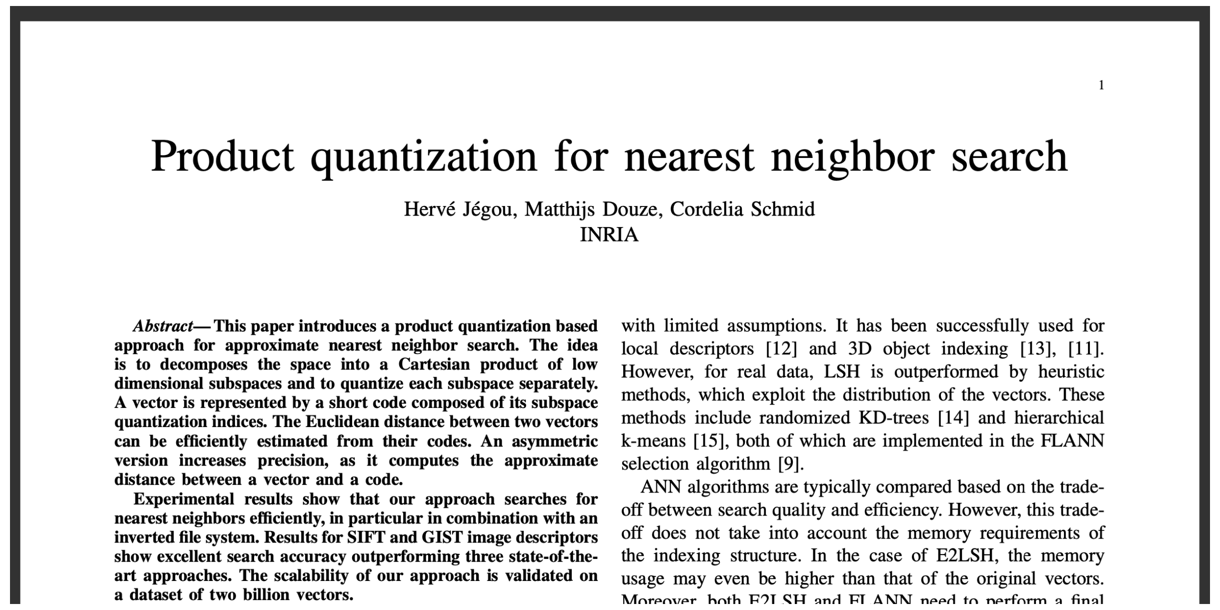
Highly recommend to watch this
For comprehensive survey
As of June 2020

Yusuke Matsui

Lecturer (Assistant Professor)

Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo, Japan

I am working as a Lecturer (Assistant Professor) at the University of Tokyo, Japan. My research is in the fields of computer vision and multimedia processing, with particular interests in large-scale indexing.



Product quantization for nearest neighbor search

Hervé Jégou, Matthijs Douze, Cordelia Schmid
INRIA

Abstract— **This paper introduces a product quantization based approach for approximate nearest neighbor search. The idea is to decomposes the space into a Cartesian product of low dimensional subspaces and to quantize each subspace separately. A vector is represented by a short code composed of its subspace quantization indices. The Euclidean distance between two vectors can be efficiently estimated from their codes. An asymmetric version increases precision, as it computes the approximate distance between a vector and a code.**

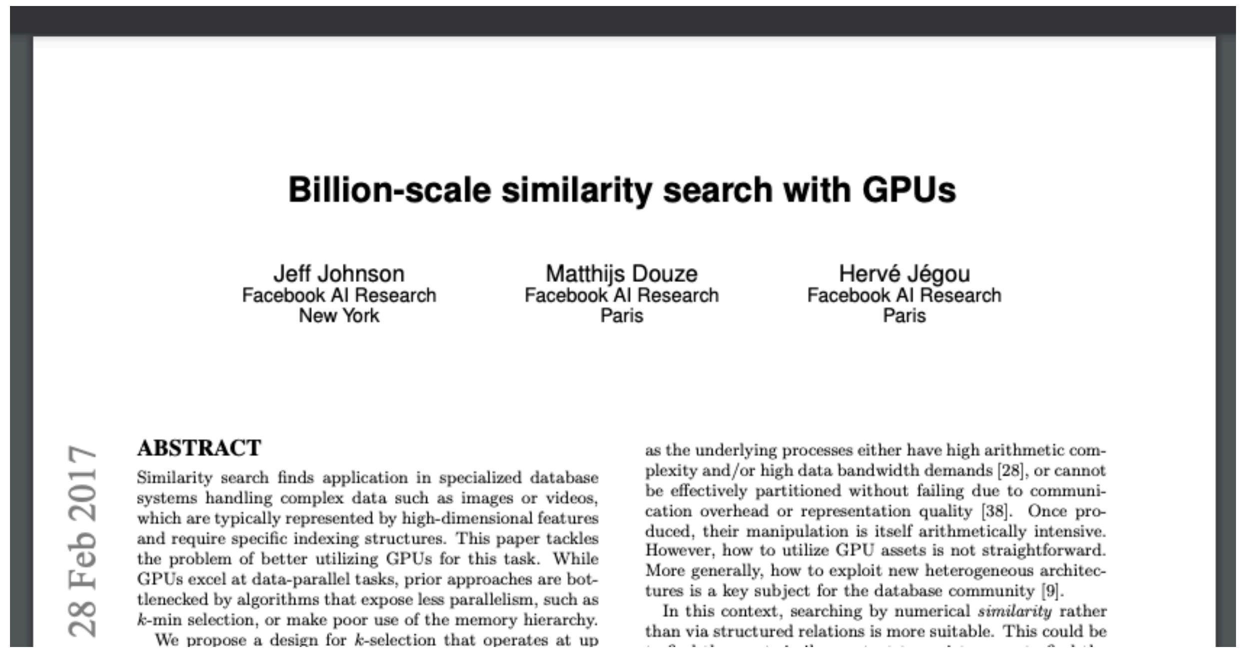
Experimental results show that our approach searches for nearest neighbors efficiently, in particular in combination with an inverted file system. Results for SIFT and GIST image descriptors show excellent search accuracy outperforming three state-of-the-art approaches. The scalability of our approach is validated on a dataset of two billion vectors.

with limited assumptions. It has been successfully used for local descriptors [12] and 3D object indexing [13], [11]. However, for real data, LSH is outperformed by heuristic methods, which exploit the distribution of the vectors. These methods include randomized KD-trees [14] and hierarchical k-means [15], both of which are implemented in the FLANN selection algorithm [9].

ANN algorithms are typically compared based on the trade-off between search quality and efficiency. However, this trade-off does not take into account the memory requirements of the indexing structure. In the case of E2LSH, the memory usage may even be higher than that of the original vectors.

<- PQ (Product Quantization)
Theory & experiments
2013

Talk about some of the contents
As prerequisites



Billion-scale similarity search with GPUs

Jeff Johnson, Facebook AI Research, New York
Matthijs Douze, Facebook AI Research, Paris
Hervé Jégou, Facebook AI Research, Paris

28 Feb 2017

ABSTRACT

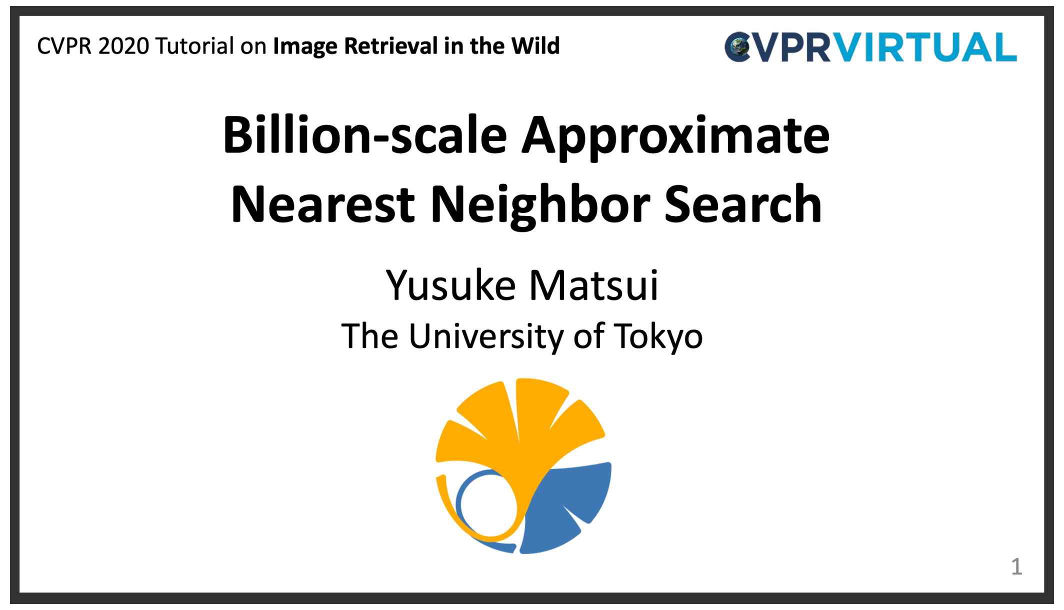
Similarity search finds application in specialized database systems handling complex data such as images or videos, which are typically represented by high-dimensional features and require specific indexing structures. This paper tackles the problem of better utilizing GPUs for this task. While GPUs excel at data-parallel tasks, prior approaches are bottlenecked by algorithms that expose less parallelism, such as k-min selection, or make poor use of the memory hierarchy.

as the underlying processes either have high arithmetic complexity and/or high data bandwidth demands [28], or cannot be effectively partitioned without failing due to communication overhead or representation quality [38]. Once produced, their manipulation is itself arithmetically intensive. However, how to utilize GPU assets is not straightforward. More generally, how to exploit new heterogeneous architectures is a key subject for the database community [9].

In this context, searching by numerical *similarity* rather than via structured relations is more suitable.

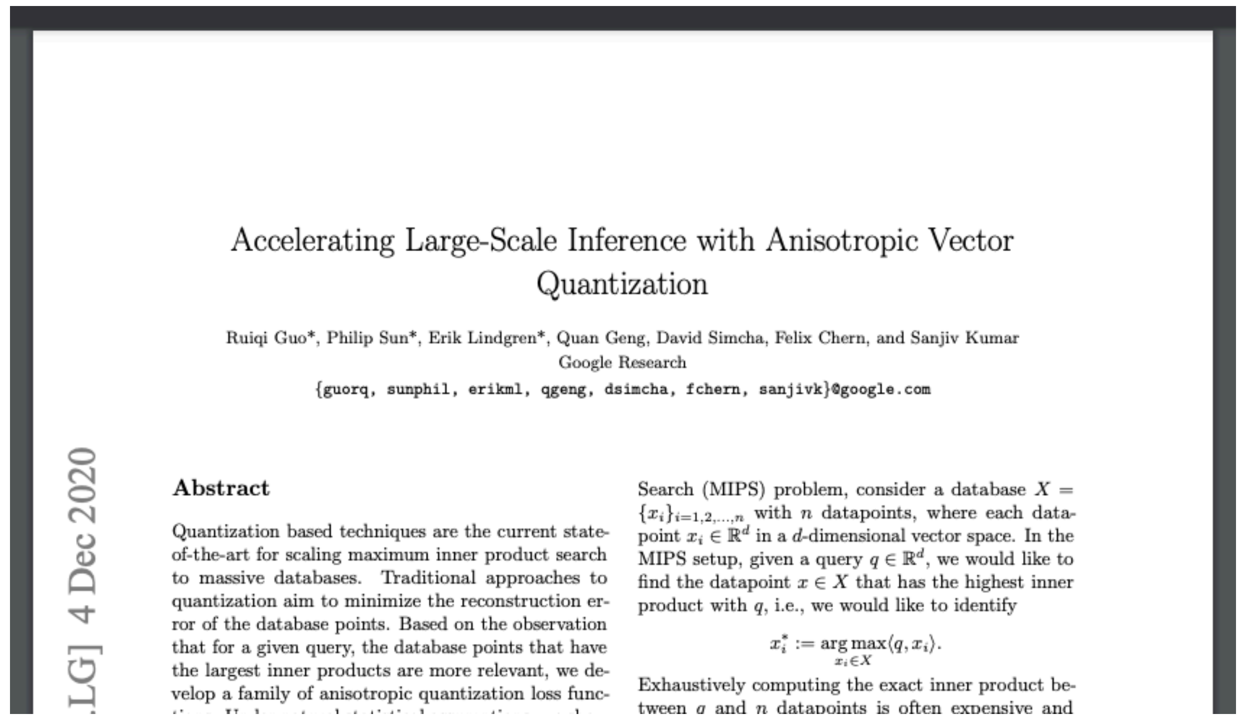
<- paper for Faiss at 2017
PQ for GPU
By PQ author & GPU expert

I couldn't understand the GPU part… :pien:



<- Slide

Some of the pages are
Copy-and-pasted in prerequisites



Accelerating Large-Scale Inference with Anisotropic Vector Quantization

Ruiqi Guo*, Philip Sun*, Erik Lindgren*, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar
Google Research
{guorq, sunphil, erikml, qgeng, dsimcha, fchern, sanjivk}@google.com

LG] 4 Dec 2020

Abstract

Quantization based techniques are the current state-of-the-art for scaling maximum inner product search to massive databases. Traditional approaches to quantization aim to minimize the reconstruction error of the database points. Based on the observation that for a given query, the database points that have the largest inner products are more relevant, we develop a family of anisotropic quantization loss func-

Search (MIPS) problem, consider a database $X = \{x_i\}_{i=1,2,\dots,n}$ with n datapoints, where each datapoint $x_i \in \mathbb{R}^d$ in a d-dimensional vector space. In the MIPS setup, given a query $q \in \mathbb{R}^d$, we would like to find the datapoint $x \in X$ that has the highest inner product with q, i.e., we would like to identify

$$x_i^* := \arg\max_{x_i \in X} \langle q, x_i \rangle.$$

Exhaustively computing the exact inner product between q and n datapoints is often expensive and

<- paper for scaNN
July 2020

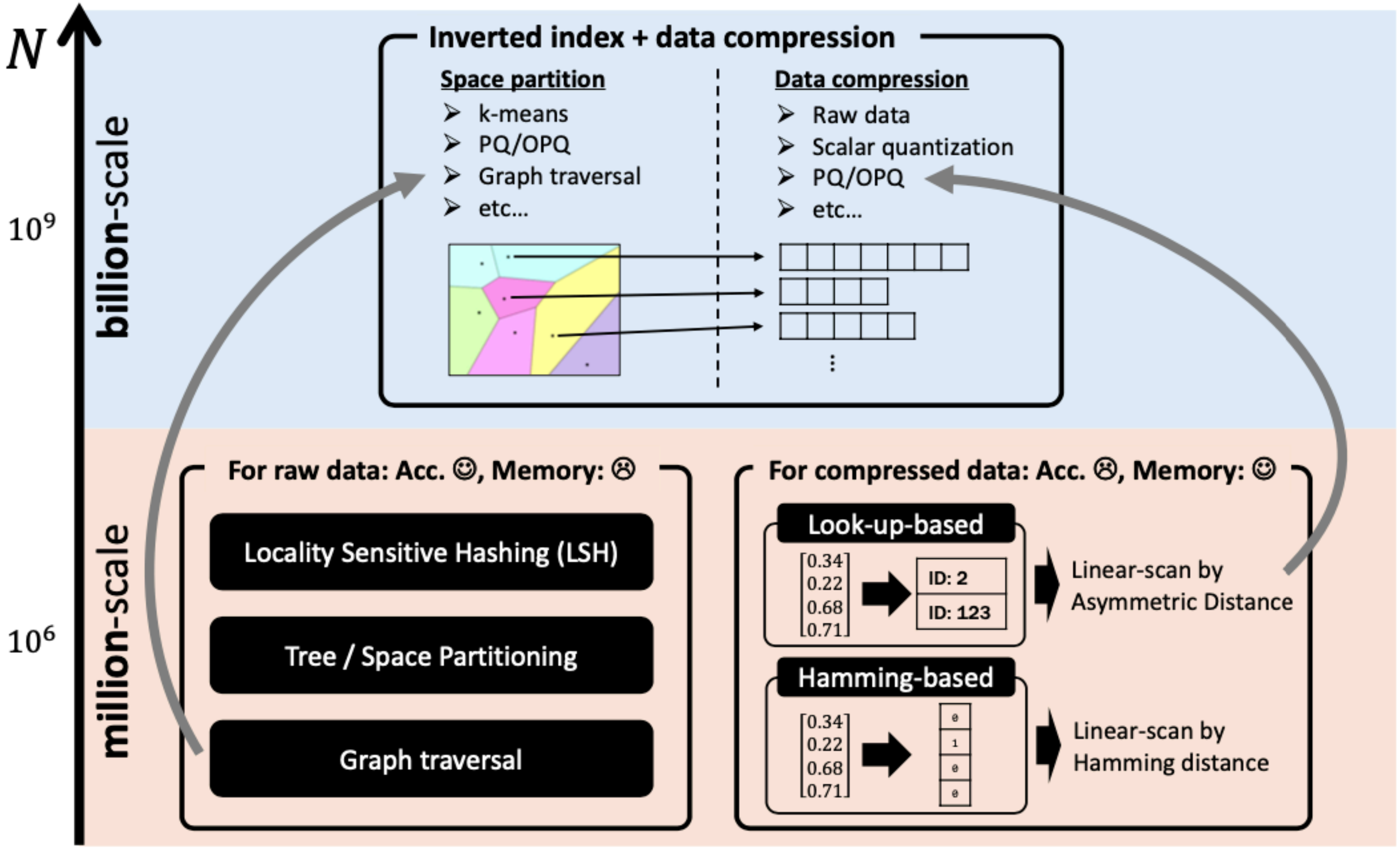
Today's topic

I. Introduction
II. Prerequisites
III. Main Topic

Prerequisites for scaNN

1. Overview of NN Algorithms
2. Product Quantization (PQ) as the best compression method
3. Inverted Index + PQ

1. Overview of NN Algorithms



Taken from the Matsui-san slide

Prerequisites for scaNN

1. Overview of NN Algorithms
2. Product Quantization (PQ) as the best compression method
3. Inverted Index + PQ

Compression (Quantization)

- Motivation: Cannot store all vectors $X = \left\{x_i \in \mathbb{R}^D\right\}_{i=1..N}$ in RAM
- Basic idea: Convert each vector into a short-code (cordeword) so it takes up less RAM
- Goal: Define $c : \mathbb{R}^D \to C$ s.t. dist$(q, x) \sim$ dist$(q, c(x))$ has the following properties
  - Less space & time cpx
  - The error $\mathbb{E}_q \left[\frac{1}{N} \sum_i \mathrm{dist}(q, x_i) - \mathrm{dist}(q, c(x_i) \right]$ should be small
- (Dimensional reduction satisfies above but is often thought of as a mere preprocess)

E.g. Naive compression by k-means

- $c : \mathbb{R}^D \to C = \left\{ c_j \text{ (centroid)} \in \mathbb{R}^D \right\}_{j=1,\cdots,K}$ by just k-means clustering
    - $\mathcal{O}(K)$ sample from X is enough for construction
- dist$(q, x) \sim$ dist(q, c_j)
    - Time cpx is still $\mathcal{O}(D)$
- Space cpx: $\mathcal{O}(KD + N \ln K)$
    - KD for the codebook (set of centroids)
    - $N \ln K$ for lookup table

- Tradeoff
    - K is small => better cpx, worse acc
    - K is large => worse cpx, better acc
- For reasonable acc, K should be large enough
    - K-means for such K is unrealistic

Product Quantization

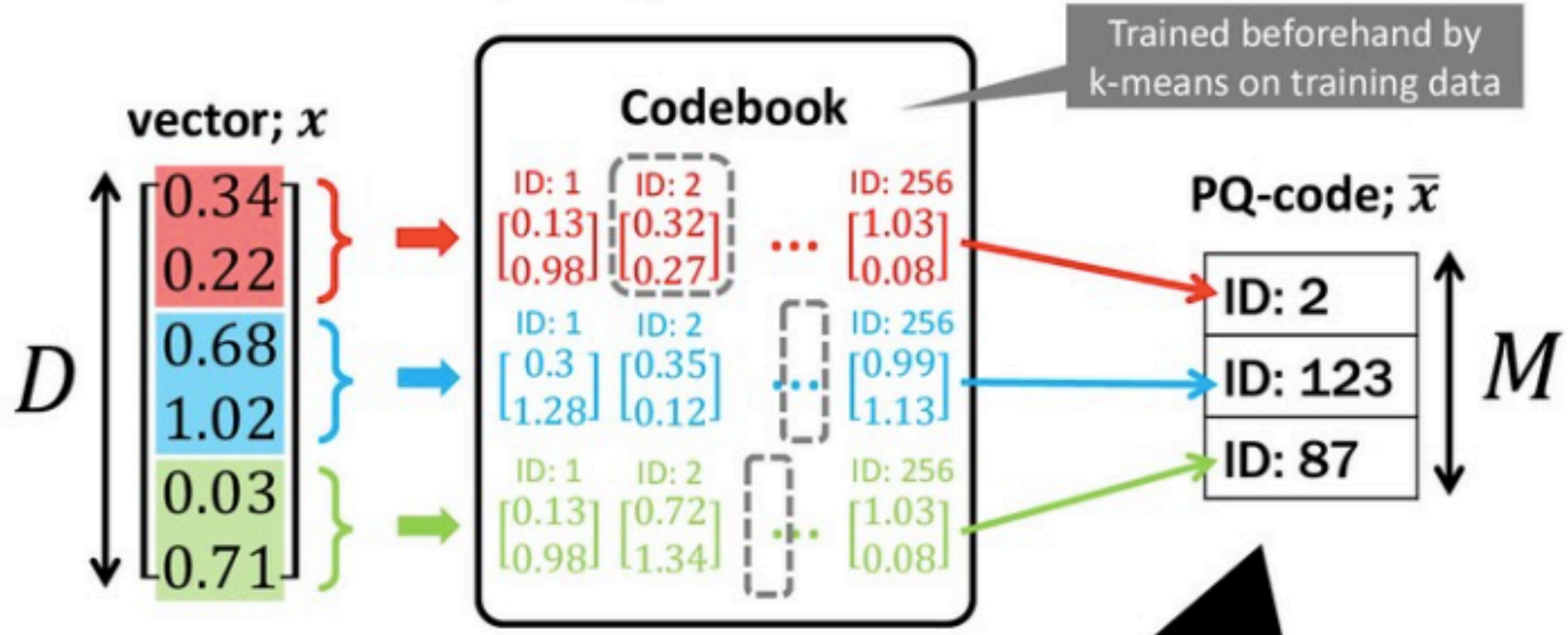
- Divide D dim into M sectors
  - Each sector has $D* = D/M$ dim
- Sub-quantize each sector
  - $c* : \mathbb{R}^{D*} \to C* = \left\{ c_j \in \mathbb{R}^{D*} \right\}_{j=1,\cdots,K*}$
- Total
  - $C = C_1 \times \cdots \times C_M$
  - $|C| = K = (K*)^M$
  - $M = 8,\ K* = 256$ has enough acc

- $\text{dist}(q, x) \sim \text{dist}(q, c)$
  - Time cpx is $\mathcal{O}(MD* = D)$
  - ADC is better than SDC
    - Asymmetric Distance Computation: use q directly
    - Symmetric Distance computation: convert q to nearest centroids
- Space cpx: $\mathcal{O}(K*D + NM \ln K*)$
  - $MK*D* = K*D$ for the codebook (K* D*-dim centroids for M sectors)
  - $NM \ln K*$ for lookup table

*Sanko- Kopipe

Product Quantization; PQ [Jégou, TPAMI 2011]

➢ Split a vector into sub-vectors, and quantize each sub-vector



➢ Simple

➢ Memory efficient

➢ Distance can be esimated

Bar notation for PQ-code in this tutorial:

$$x \in \mathbb{R}^D \mapsto \bar{x} \in \{1, \ldots, 256\}^M$$

Prerequisites for scaNN

1. Overview of NN Algorithms
2. Product Quantization (PQ) as the best compression method
3. Inverted Index + PQ to find the closest quickly

Inverted Index

ID	w1	w2	w3	w4	...
1	I	love	you	so	…
2	I	am	a	cat	…
N	You	are	a	dog	…

↓

w	ID1	ID2	...
I	1	2	…
love	1	15	…
you	1	…	N
am	2	…	
a	2	…	
cat	2	…	

K

- For given query (= search word)
  - Find the word from the index $\mathcal{O}(\ln K \text{ or } 1)$
  - Return the document IDs

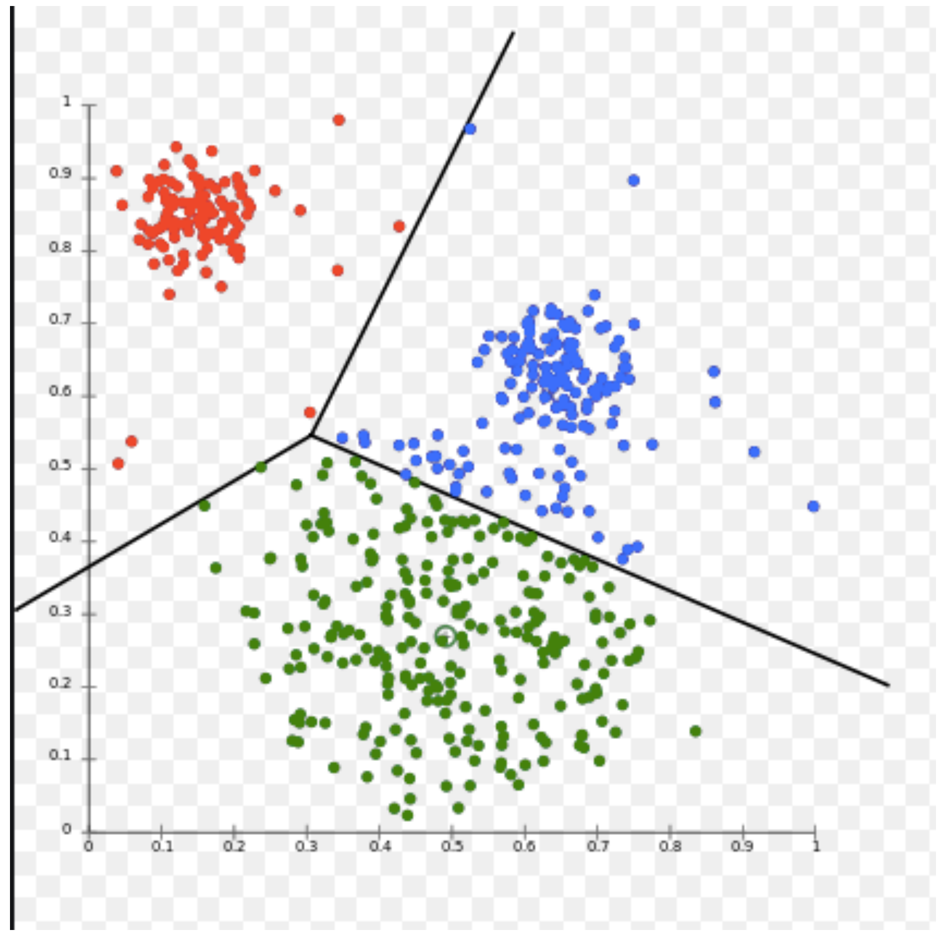
Inverted Index for NN

$$X = \left\{x_i \in \mathbb{R}^D\right\}_{i=1,\cdots,N}$$

	ID1	ID2	...
c1	1	3	...
c2	4	15	...
cK	9	28	...

$$\mathcal{V}_j = \left\{ i = 1,\cdots,N \mid c(x_i) = c_j \right\}$$

- For given query vector
  - Find the τ-nearest centroids
    - Linear search: $\mathcal{O}(KD)$
    - Better (e.g. HNSW)
  - Find $\mathrm{argmin}_{i \in \mathcal{V}_{j=1,\cdots,\tau}} \mathrm{dist}(q, x_i)$



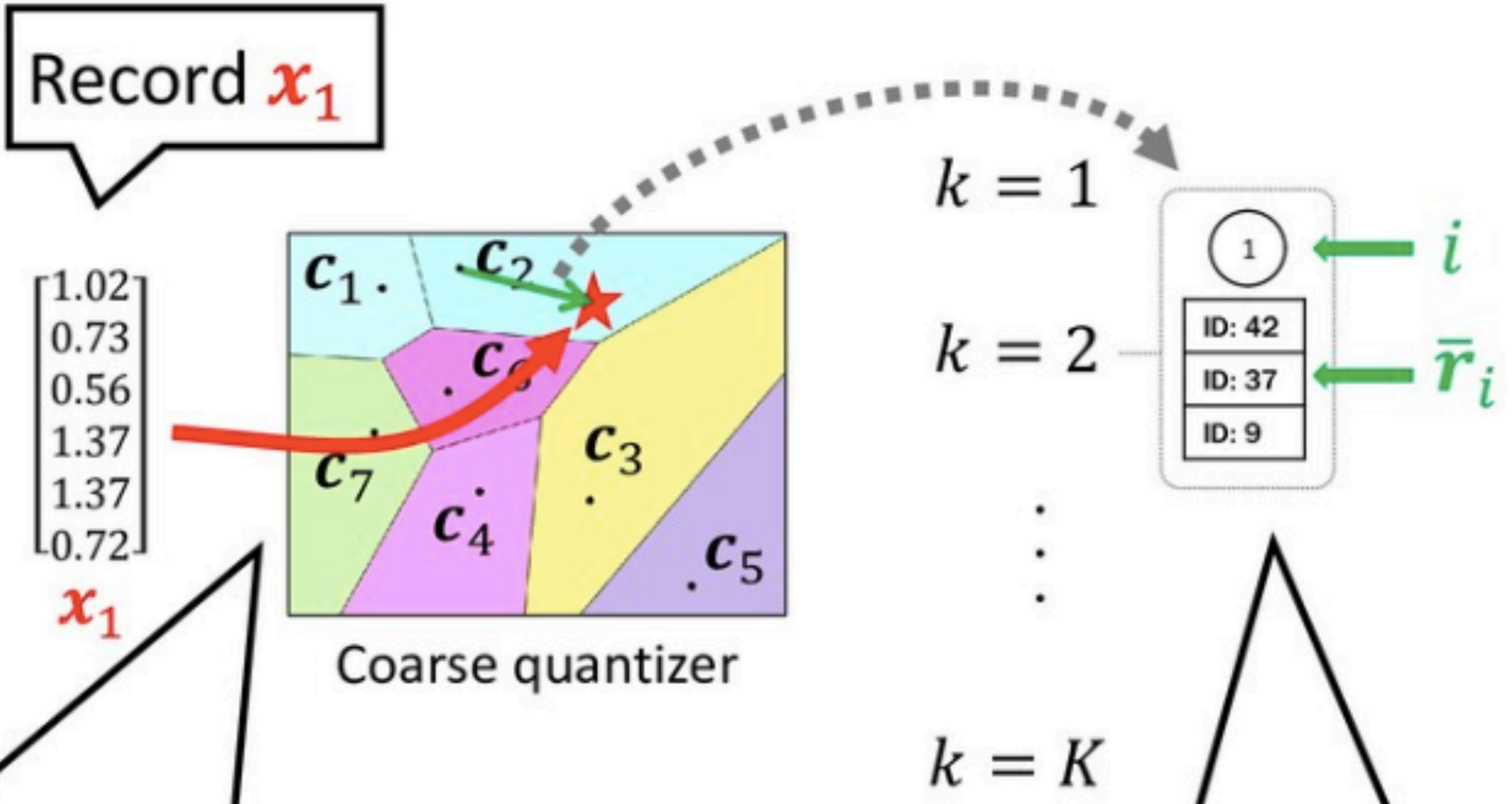
Inverted Index + PQ

- Called IVFADC
  (InVerted File system with Asymmetric Distance Computation)
- $c : x \mapsto c(x) = c_c(x) + c_f(r := x - c_c(x))$
  - c_c : coarse quantizer (K'-means)
  - c_f : fine quantizer (PQ)

	ID1	ID2	…
c1	(1, cf(r1))	(3, cf(r3))	…
c2	(4, cf(r4))	(15, cf(r15))	…
cK'	(9, cf(r9))	(28, cf(r28))	…

- Typically
  $K' \sim \sqrt{N},\ M = 8,\ K^* = 2^8$
- Space cpx:
  $\mathcal{O}(K'D + K^*D + N(\ln K + \ln K^*))$
- Time cpx for search:
  $\mathcal{O}(K'D + \tau N/K'D)$

Inverted index + PQ: Record



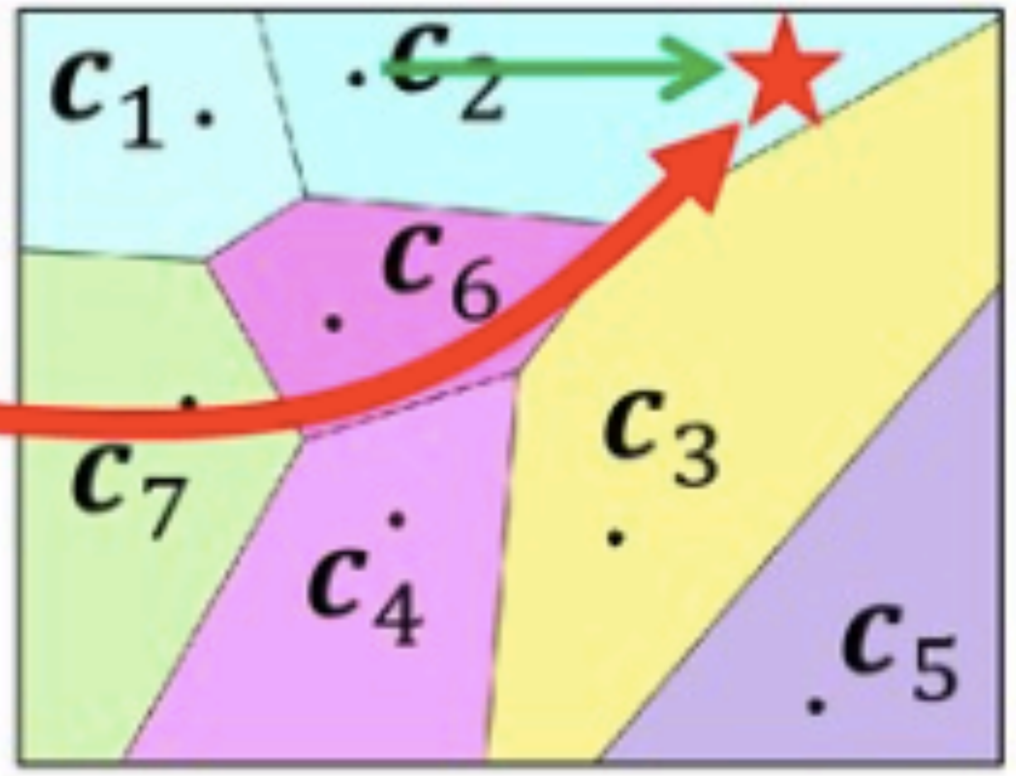
- $\boldsymbol{c}_2$ is closest to $\boldsymbol{x}_1$
- Compute a **residual** r_1 between $\boldsymbol{x}_1$ and $\boldsymbol{c}_2$:
  $\boldsymbol{r}_1 = \boldsymbol{x}_1 - \boldsymbol{c}_2 \quad (\longrightarrow)$

- Quantize $\boldsymbol{r}_1$ to $\bar{\boldsymbol{r}}_1$ by PQ
- Record it with the ID, “1”
- i.e., record $(i, \bar{\boldsymbol{r}}_i)$

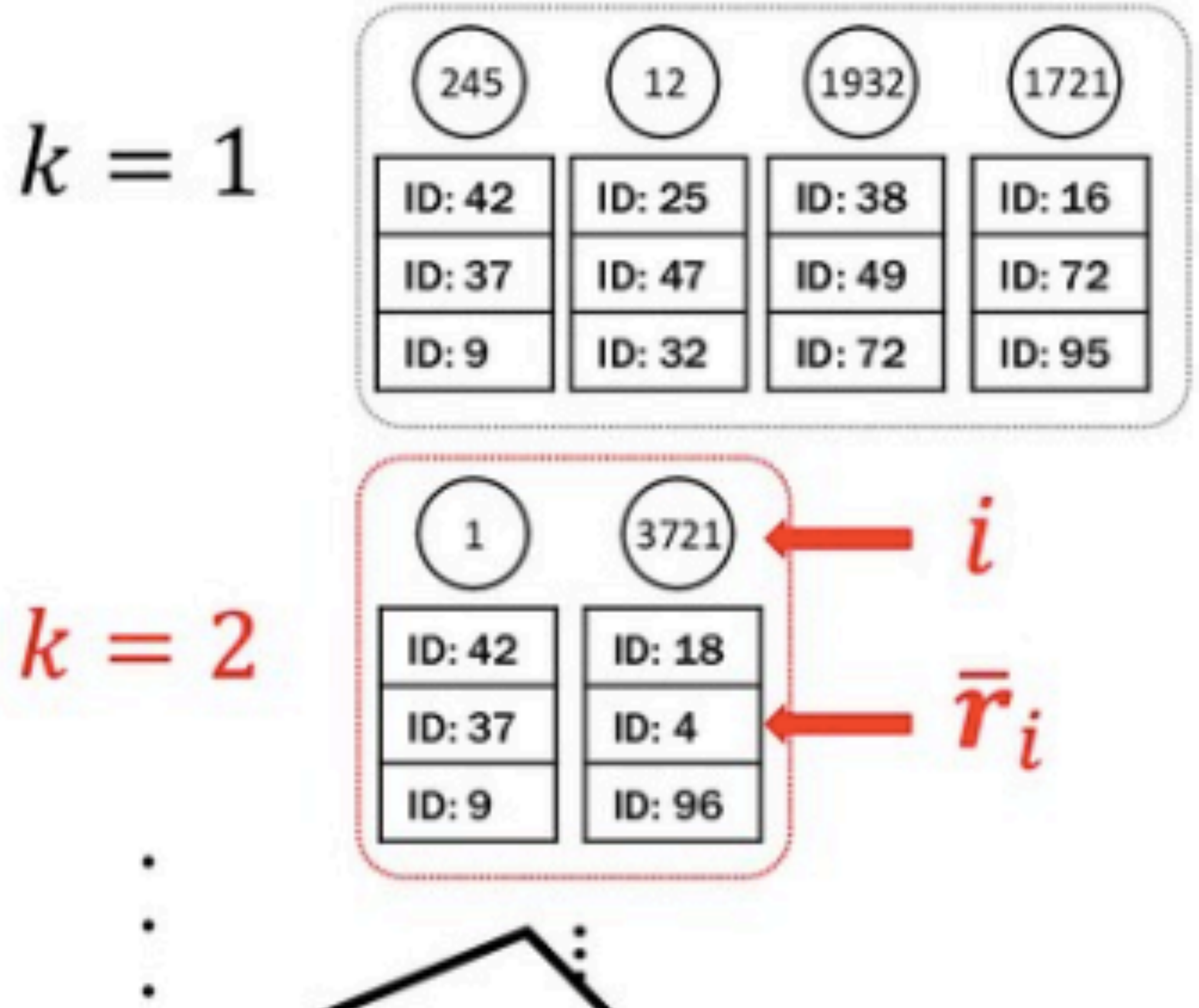
Inverted index + PQ: Search

Find the nearest vector to $\boldsymbol{q}$

$$\boldsymbol{q} = \begin{bmatrix} 0.54 \\ 2.35 \\ 0.82 \\ 0.42 \\ 0.14 \\ 0.32 \end{bmatrix}$$



coarse quantizer



- $\boldsymbol{c}_2$ is the closest to $\boldsymbol{q}$
- Compute the residual: $\boldsymbol{r}_q = \boldsymbol{q} - \boldsymbol{c}_2$

- For all $(i, \bar{\boldsymbol{r}}_i)$ in $k = 2$, compare $\bar{\boldsymbol{r}}_i$ with $\boldsymbol{r}_q$:

$$d(\boldsymbol{q}, \boldsymbol{x}_i)^2 = d(\boldsymbol{q} - \boldsymbol{c}_2, \boldsymbol{x}_i - \boldsymbol{c}_2)^2$$
$$= d(\boldsymbol{r}_q, \boldsymbol{r}_i)^2 \sim d_A(\boldsymbol{r}_q, \bar{\boldsymbol{r}}_i)^2$$

- Find the smallest one (several strategies)

I. Introduction
II. Prerequisites
III. Main Topic