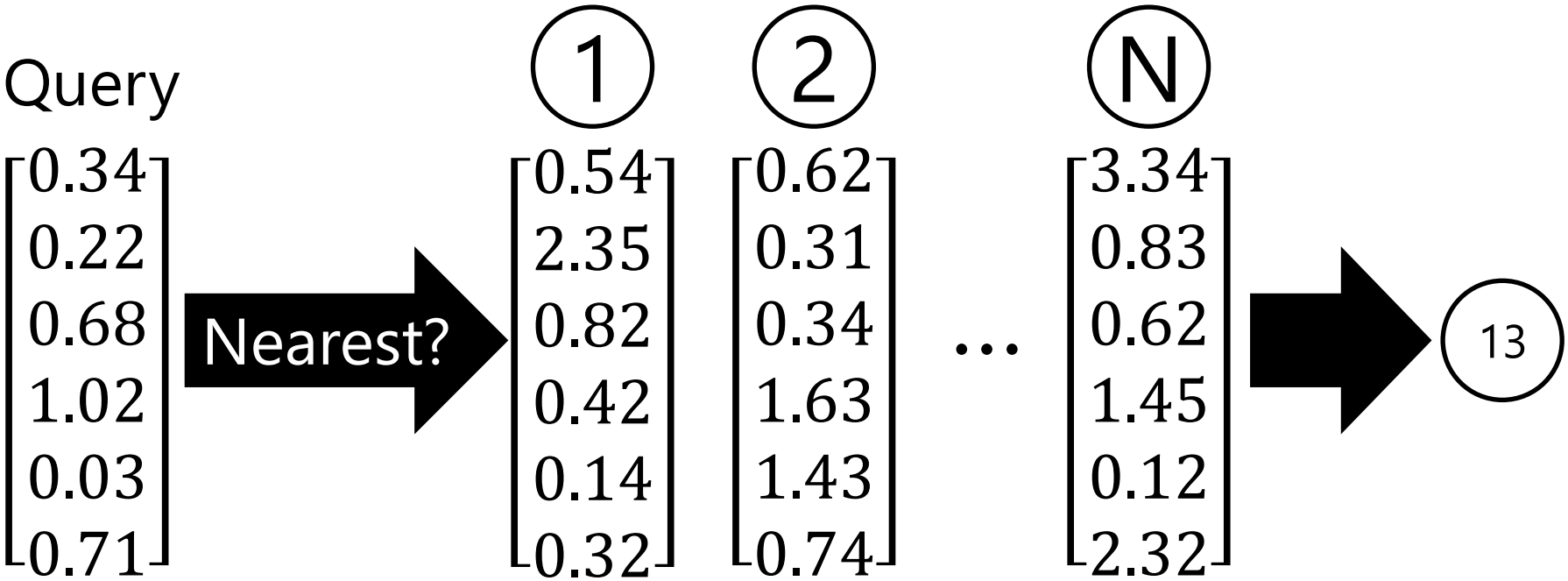
ショートコードによる
大規模近似最近傍探索

2016/12/2 @大阪大学

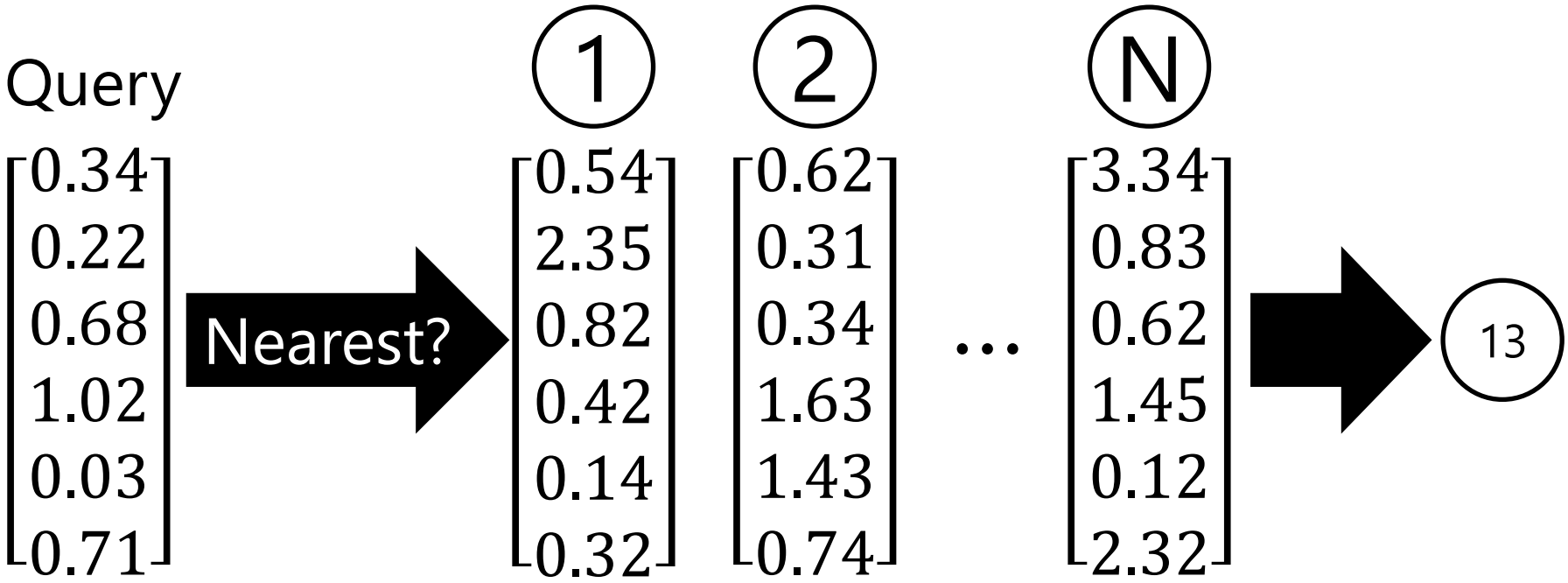
松井勇佑
国立情報学研究所

最近傍探索



- N個のベクトルがある
- クエリベクトルが与えられたとき，
  一番近いベクトルを探す
- 計算機科学における基本的な問題

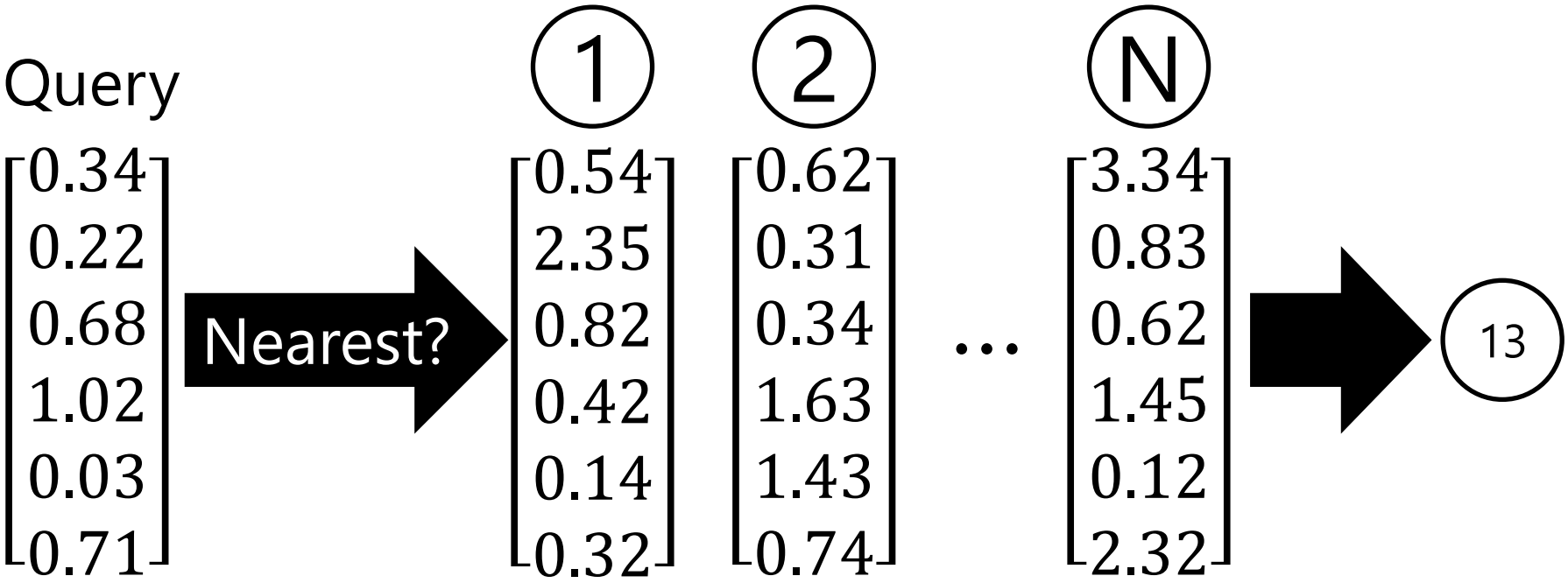
最近傍探索



$$n^* = \arg\min_{1 \le n \le N} \|\boldsymbol{q} - \boldsymbol{x}_n\|^2$$

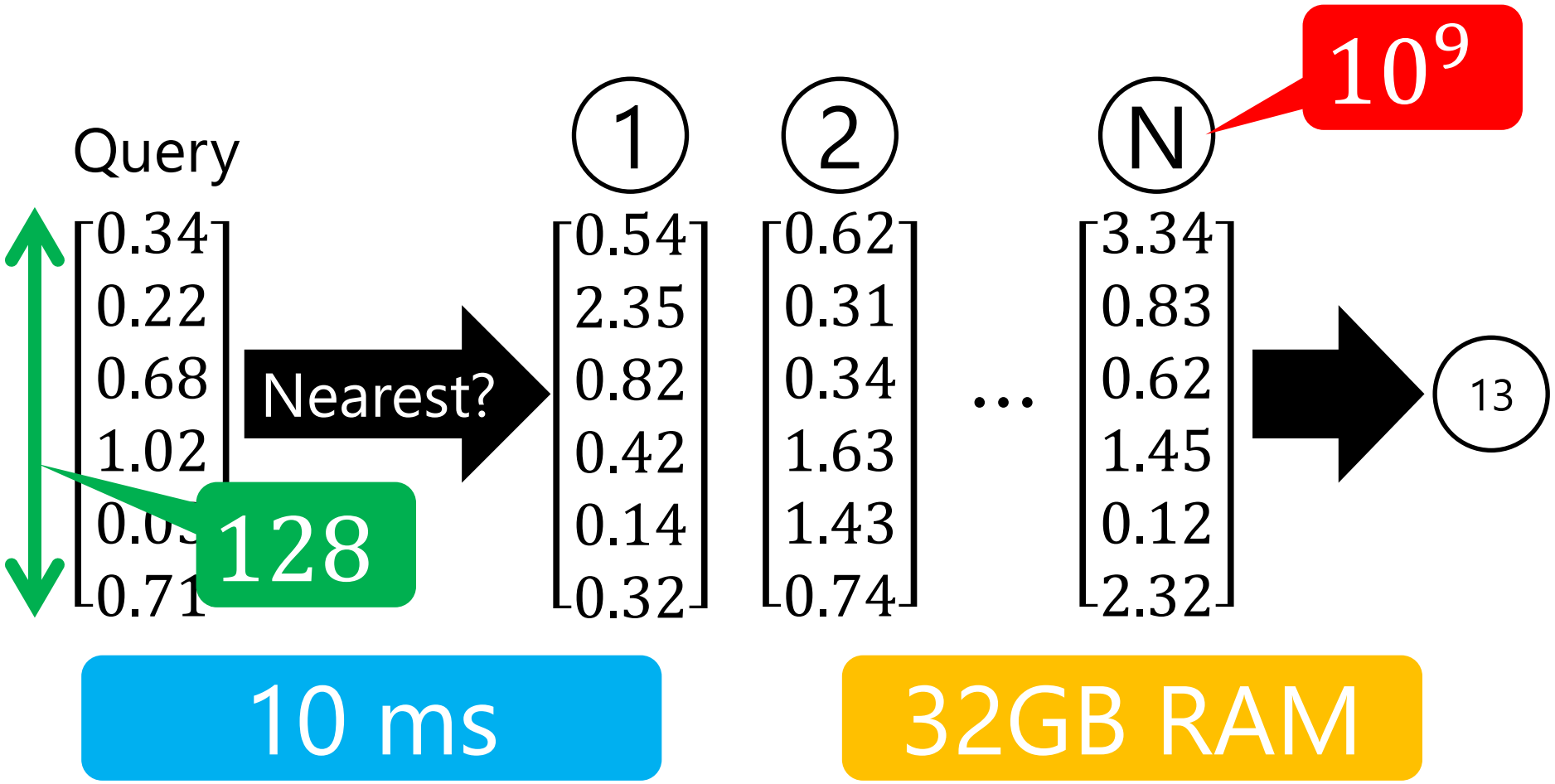
- 真面目に線形に全探索：$O(DN)$
- 遅い

近似最近傍探索



- **近似**最近傍探索
- 厳密に最近傍でなくてもよいので，
  高速に解を求める
- 速度，メモリ，精度のトレードオフ

近似最近傍探索



10 ms

32GB RAM

- 今日紹介する手法の規模感
- **大規模**な近似最近傍探索を扱う

近似最近傍探索

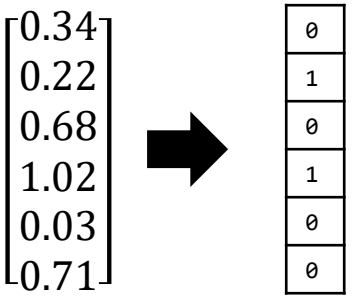
Locality Sensitive Hashing (LSH)系

Tree系
(e.g., FLANN [Muja, PAMI 14])

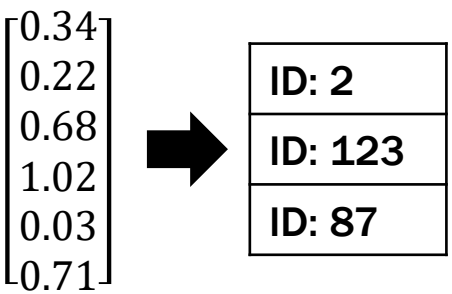
etc.

ショートコード系

ハミング系



ルックアップ系



近似最近傍探索

メモリ消費大

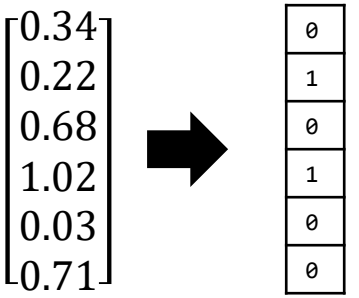
Locality Sensitive Hashing (LSH)系

(e.g., FLANN)

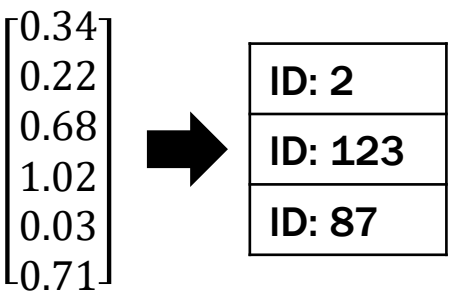
etc.

ショートコード系

ハミング系



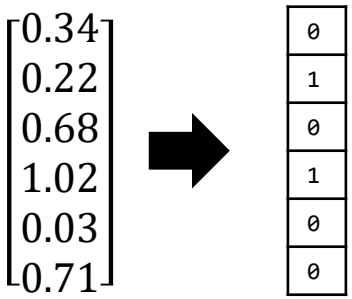
ルックアップ系



ショートコードによる近似最近傍探索

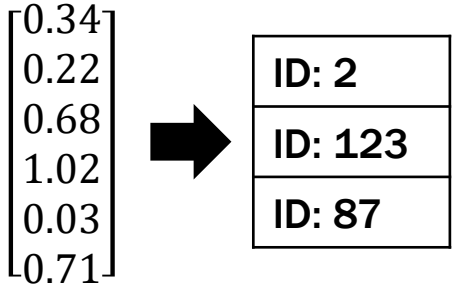
- 基本的な考え方

ハミング系



- 原理
- 手法の例
- 高速計算

ルックアップ系

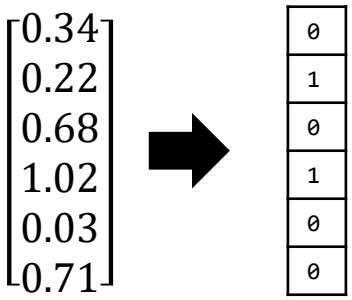


- 原理
- 探索システム

ショートコードによる近似最近傍探索

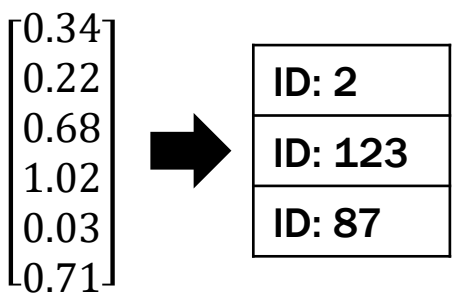
- 基本的な考え方

ハミング系



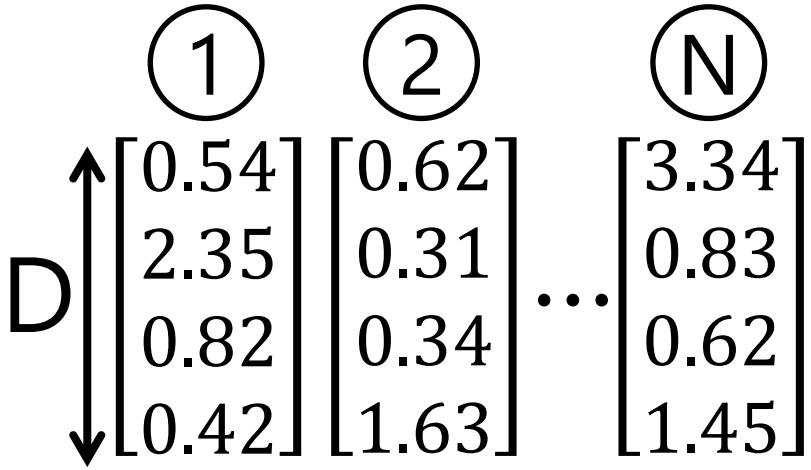
- 原理
- 手法の例
- 高速計算

ルックアップ系

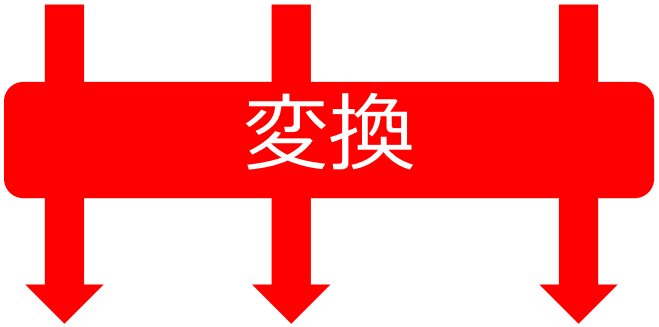


- 原理
- 探索システム

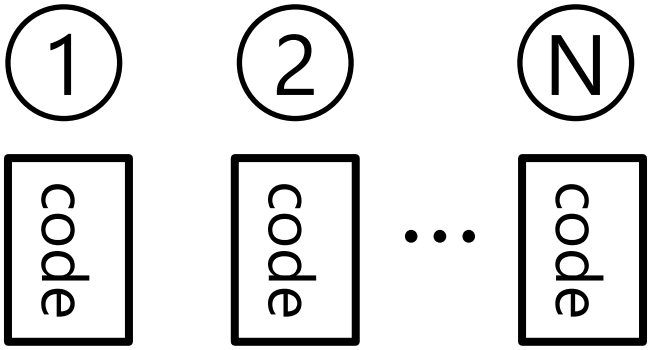
基本的な考え方



- N個の実数値ベクトルを表現するにはfloatを用いて$4ND$ byte 必要
- NやDが大きいとメモリに載らない
- 例：$D = 128, N = 10^9$の時，512 GB

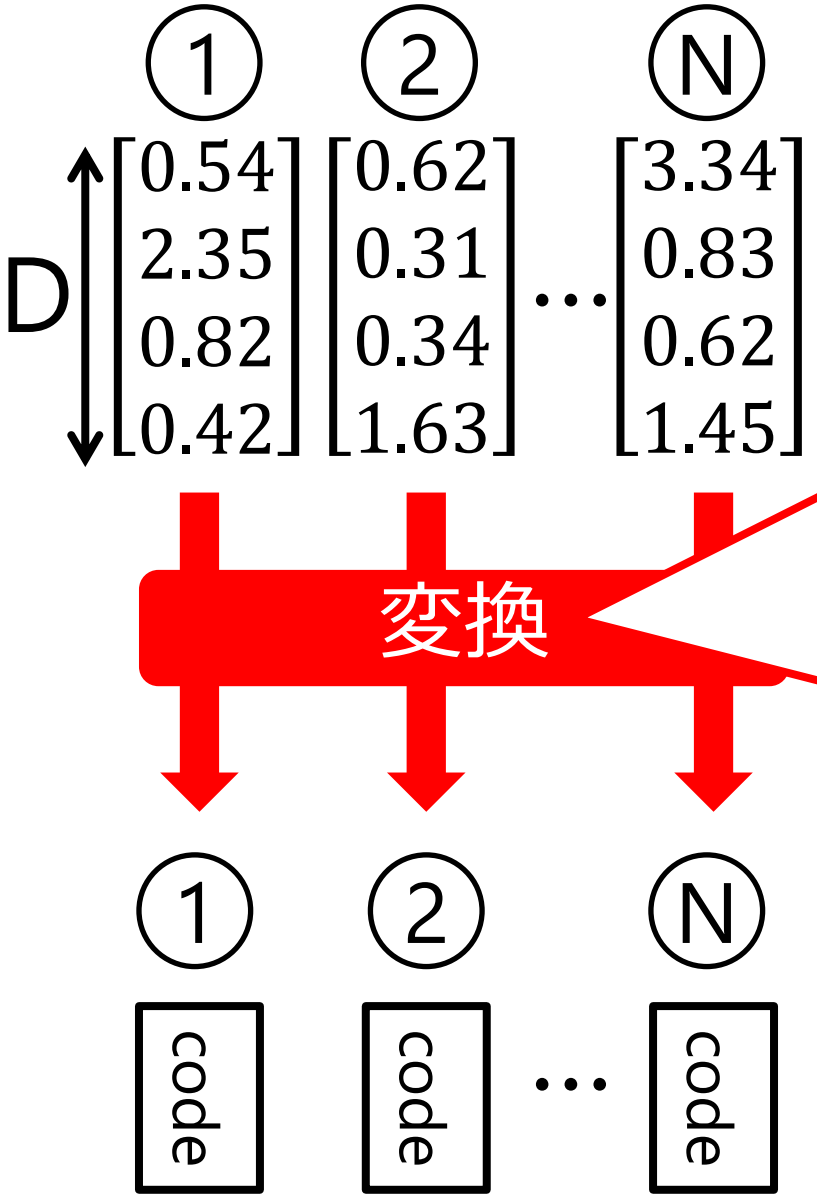


- 各ベクトル「変換」し「ショートコード」に圧縮する



- ショートコードはメモリ効率が良い
- 例：上のデータを32bitコードに圧縮すると，わずか4GB
- 以降，ショートコードの世界で探索を考える

基本的な考え方



どのような変換がいいか？

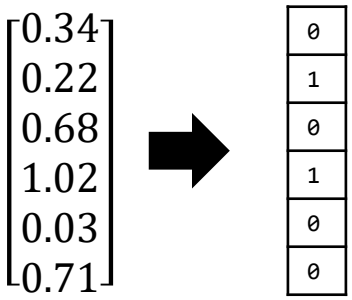
1. コード間の「距離」が計算できる．その距離は元のベクトル間の距離を近似する
2. コード間距離は高速に計算できる
3. 上の二つを，十分に小さいコードで実現できる

- メモリ効率が良い
- 例：上のデータを32bitコードに圧縮すると，わずか4GB
- 以降，ショートコードの世界で探索を考える

ショートコードによる近似最近傍探索

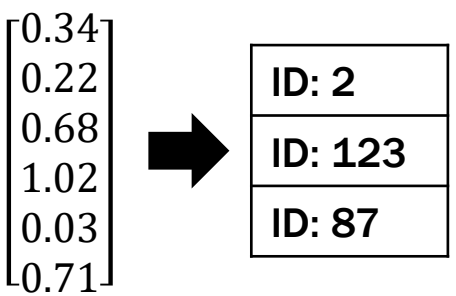
- 基本的な考え方

ハミング系



- 原理
- 手法の例
- 高速計算

ルックアップ系

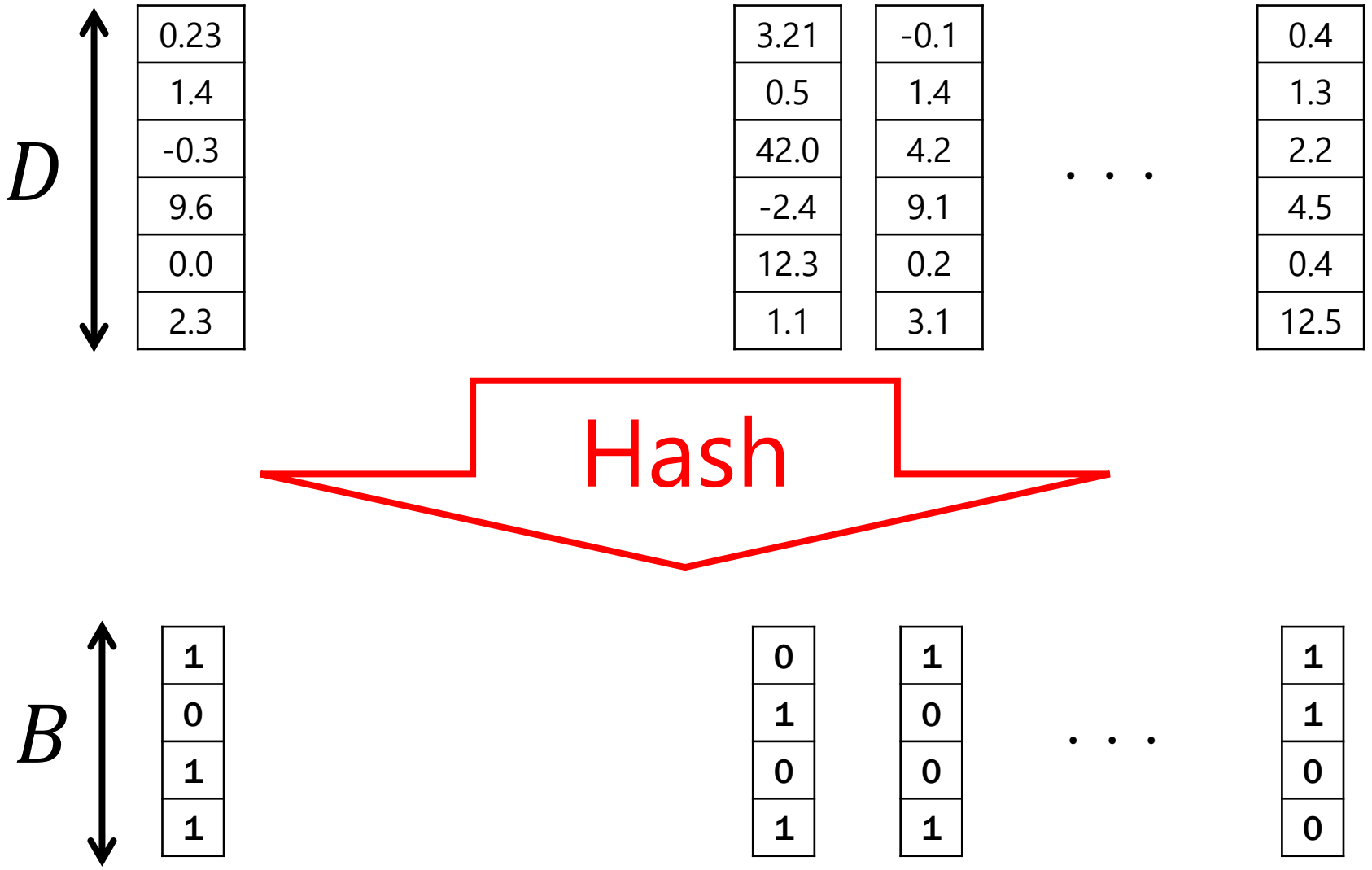


- 原理
- 探索システム

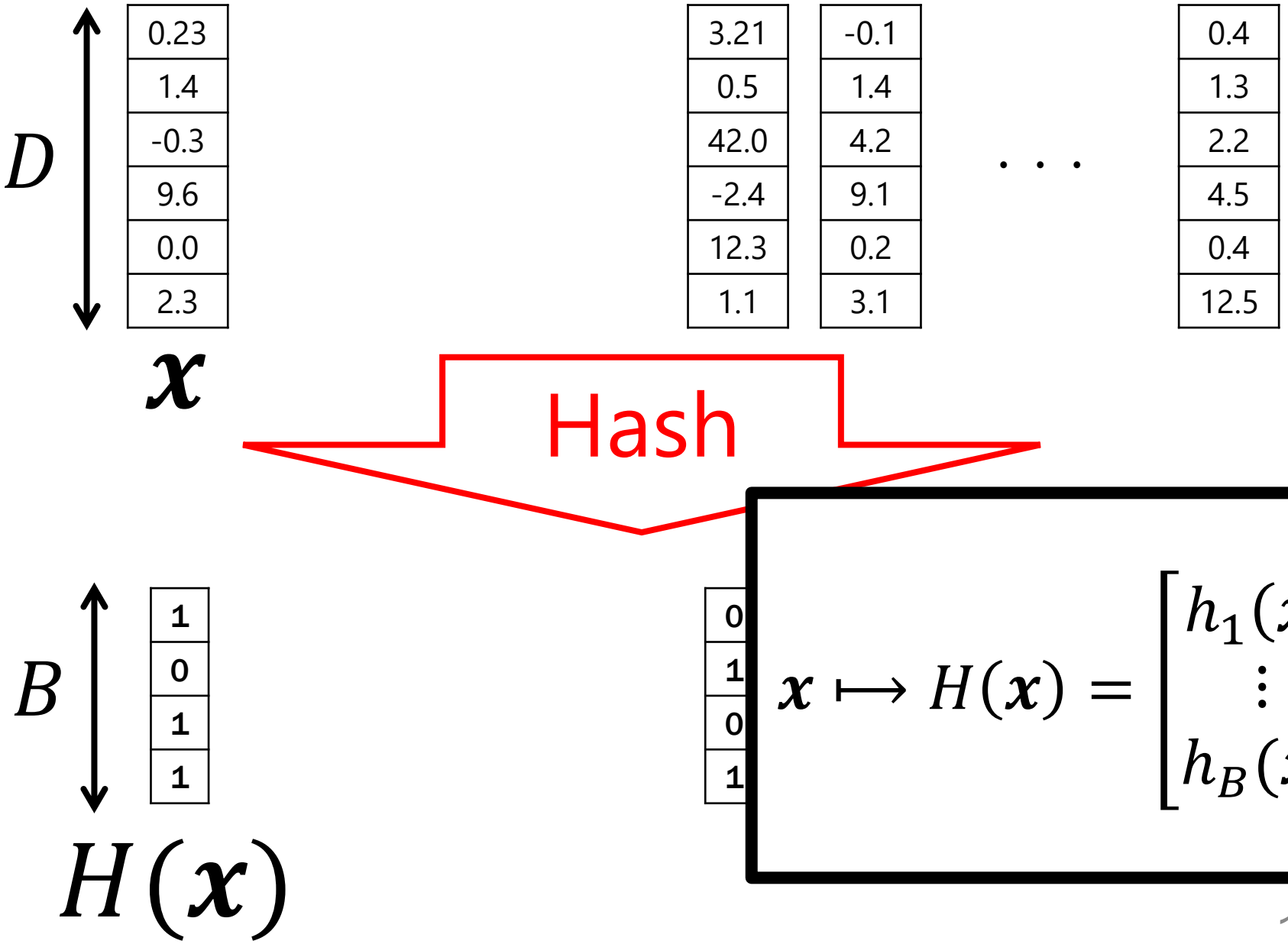
ハミング系手法：原理


D
0.23
1.4
-0.3
9.6
0.0
2.3
3.21
0.5
42.0
-2.4
12.3
1.1
-0.1
1.4
4.2
9.1
0.2
3.1
0.4
1.3
2.2
4.5
0.4
12.5

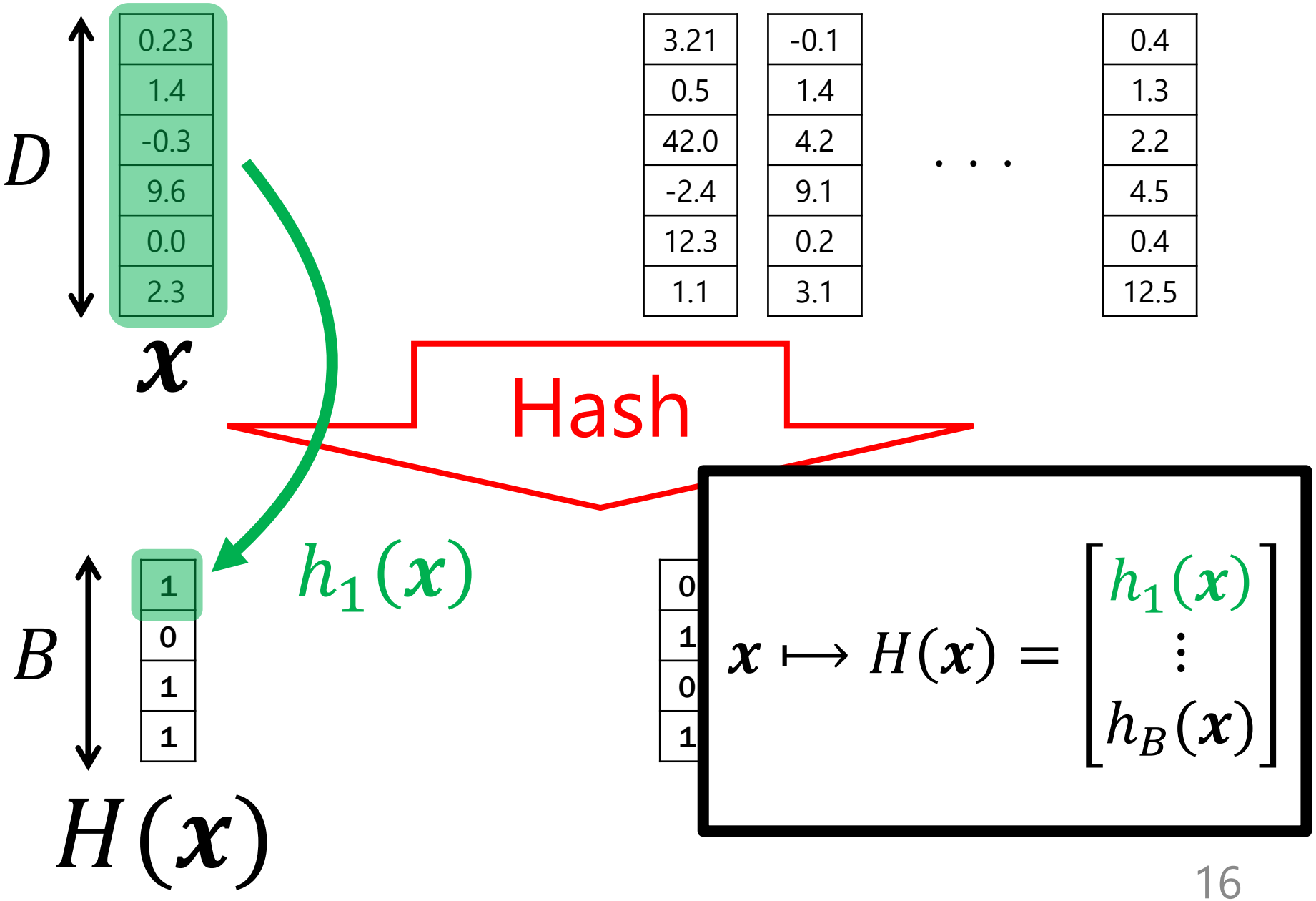
ハミング系手法：原理


D
0.23
1.4
-0.3
9.6
0.0
2.3
3.21
0.5
42.0
-2.4
12.3
1.1
-0.1
1.4
4.2
9.1
0.2
3.1
0.4
1.3
2.2
4.5
0.4
12.5
Hash
B
1
0
1
1
0
1
0
1
1
0
0
1
1
1
0
0

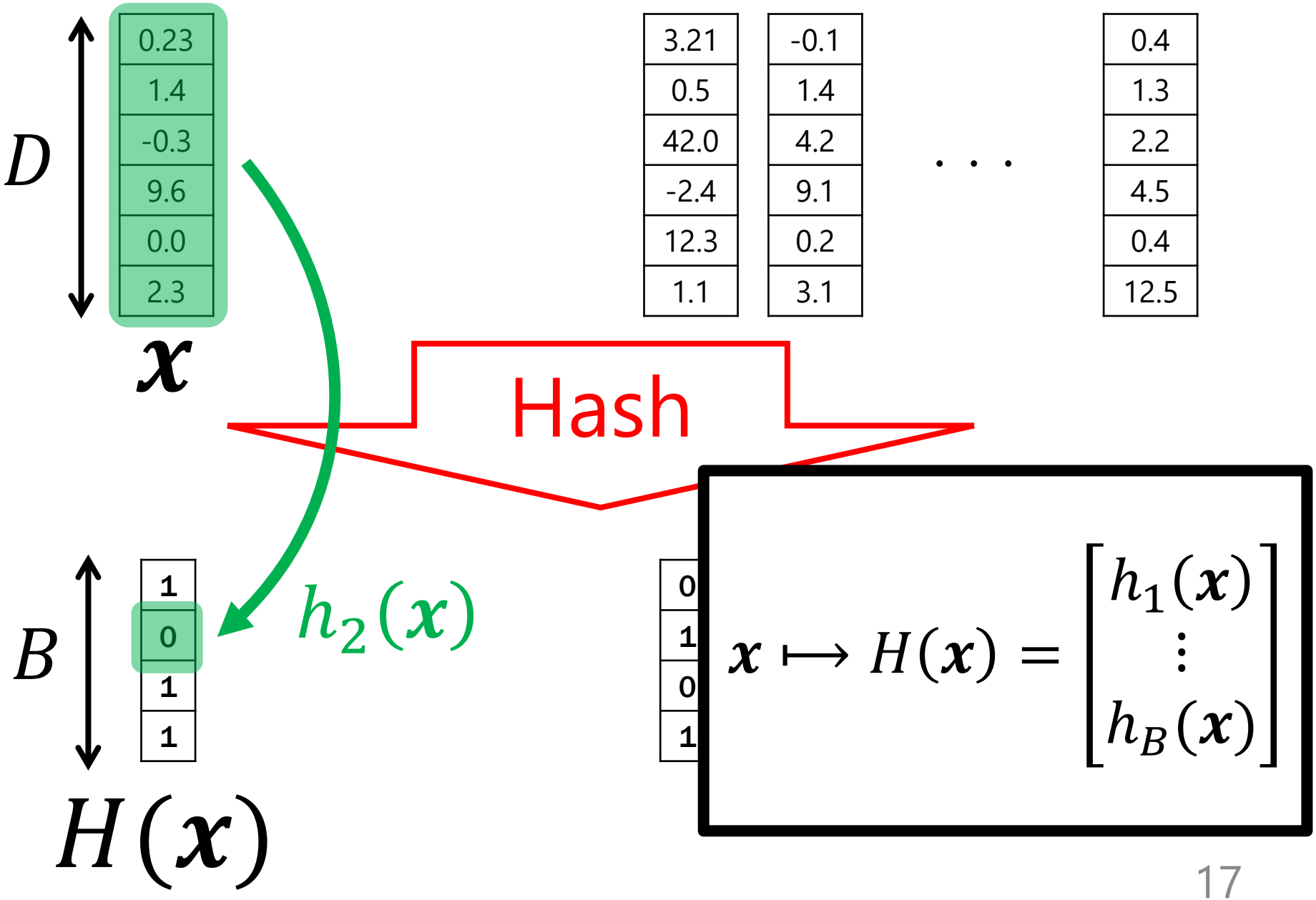
ハミング系手法：原理


D
0.23
1.4
-0.3
9.6
0.0
2.3
x
3.21
0.5
42.0
-2.4
12.3
1.1
-0.1
1.4
4.2
9.1
0.2
3.1
. . .
0.4
1.3
2.2
4.5
0.4
12.5
Hash
B
1
0
1
1
H(x)
0
1
0
1
x ⟼ H(x) = [h_1(x) ⋮ h_B(x)]

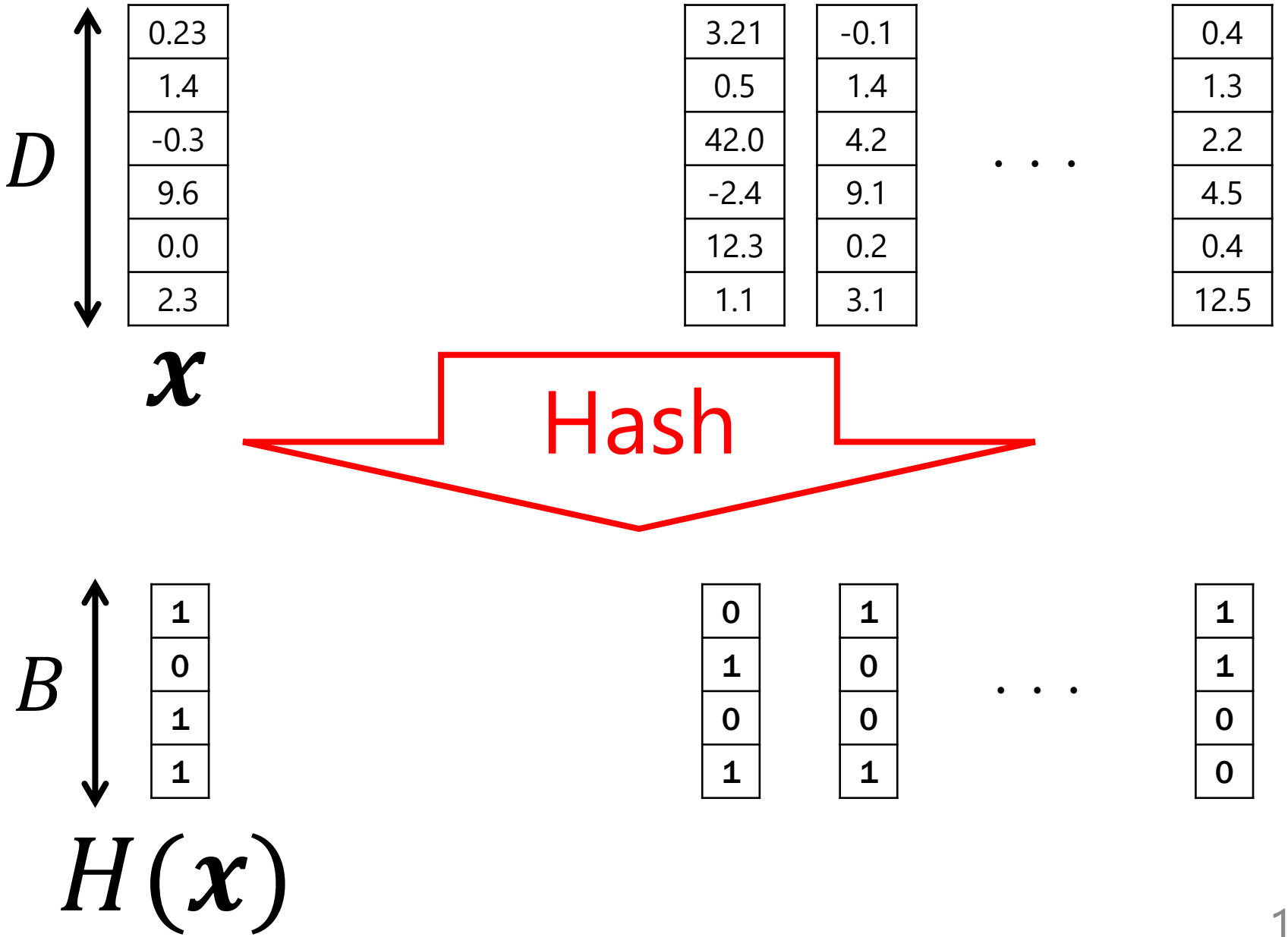
ハミング系手法：原理


D
0.23
1.4
-0.3
9.6
0.0
2.3
x
3.21
0.5
42.0
-2.4
12.3
1.1
-0.1
1.4
4.2
9.1
0.2
3.1
. . .
0.4
1.3
2.2
4.5
0.4
12.5
Hash
$h_1(x)$
B
1
0
1
1
H(x)
0
1
0
1
$x \longmapsto H(x) = \begin{bmatrix} h_1(x) \\ \vdots \\ h_B(x) \end{bmatrix}$

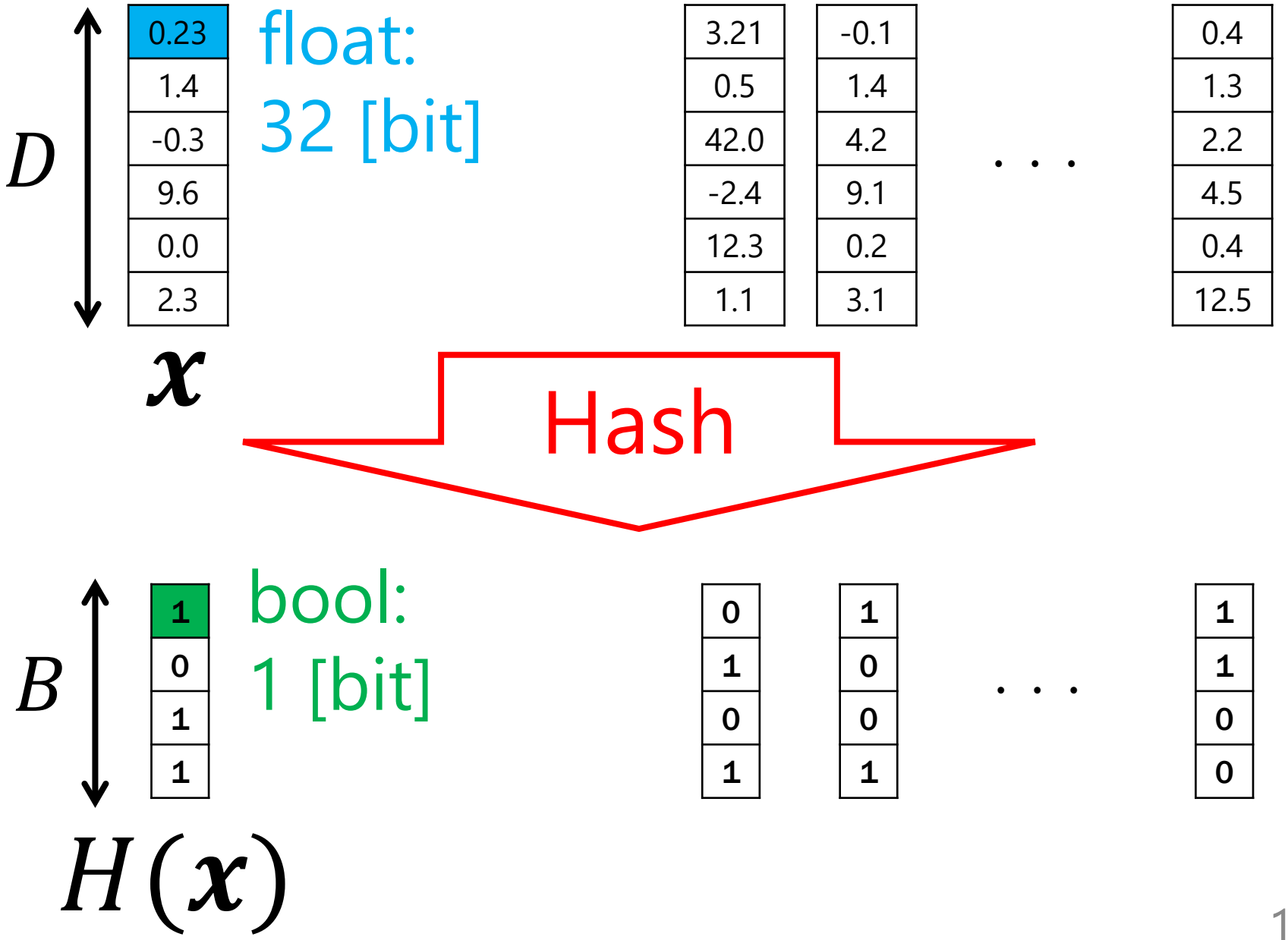
ハミング系手法：原理


D
0.23
1.4
-0.3
9.6
0.0
2.3
x
3.21
0.5
42.0
-2.4
12.3
1.1
-0.1
1.4
4.2
9.1
0.2
3.1
. . .
0.4
1.3
2.2
4.5
0.4
12.5
Hash
B
1
0
1
1
H(x)
h2(x)
0
1
0
1
x ⟼ H(x) = [h1(x) ⋮ hB(x)]

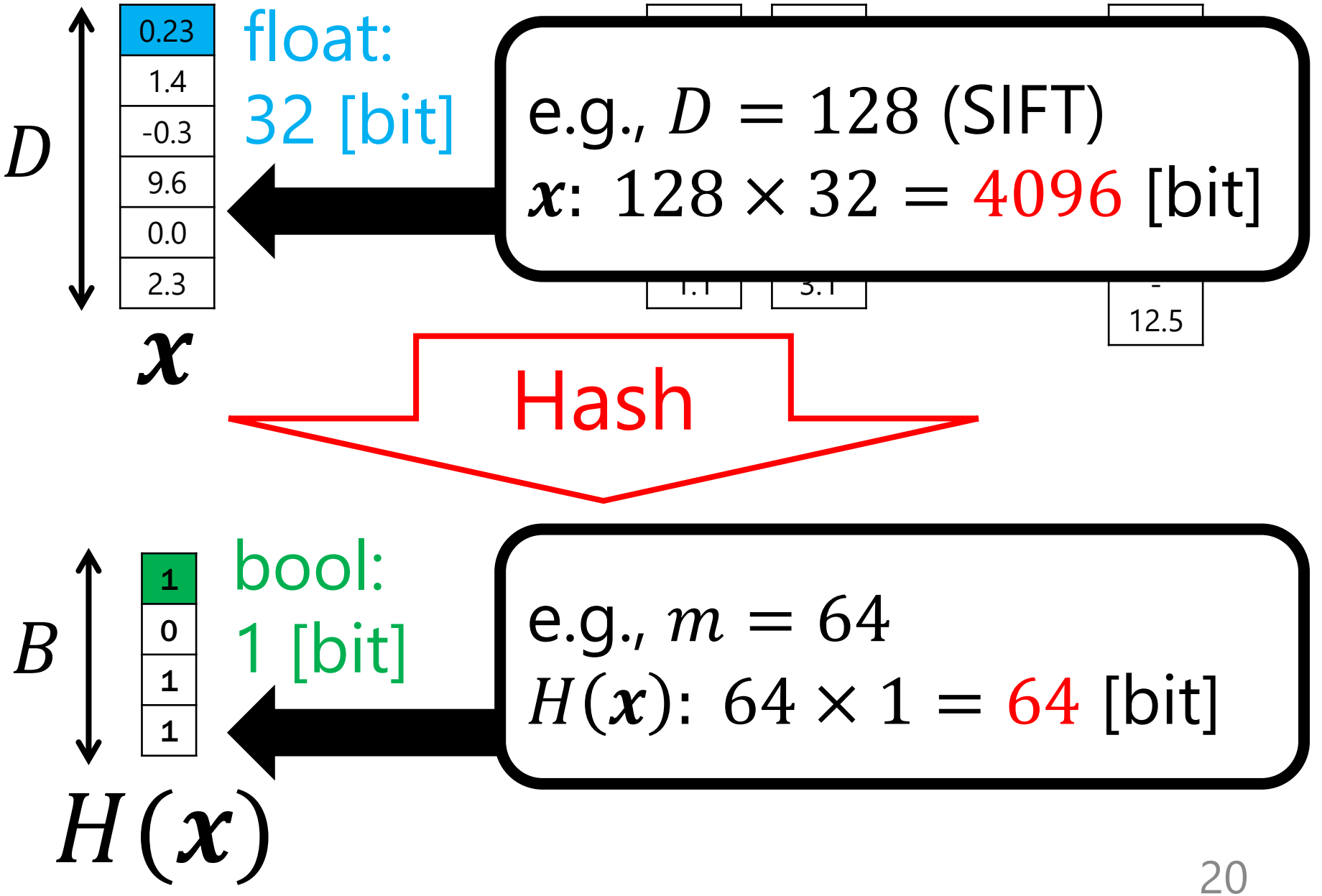
ハミング系手法：メモリ効率



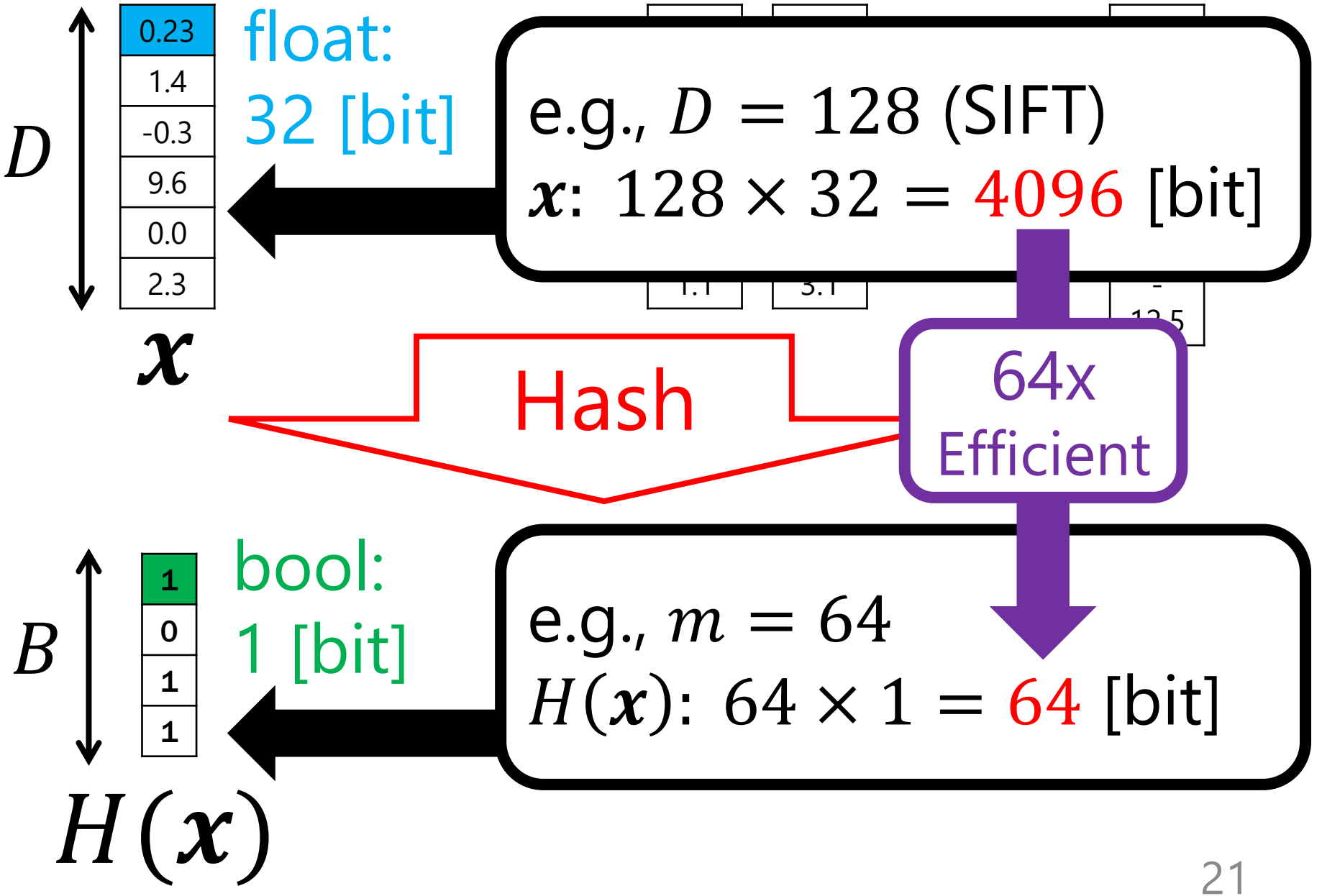
ハミング系手法：メモリ効率



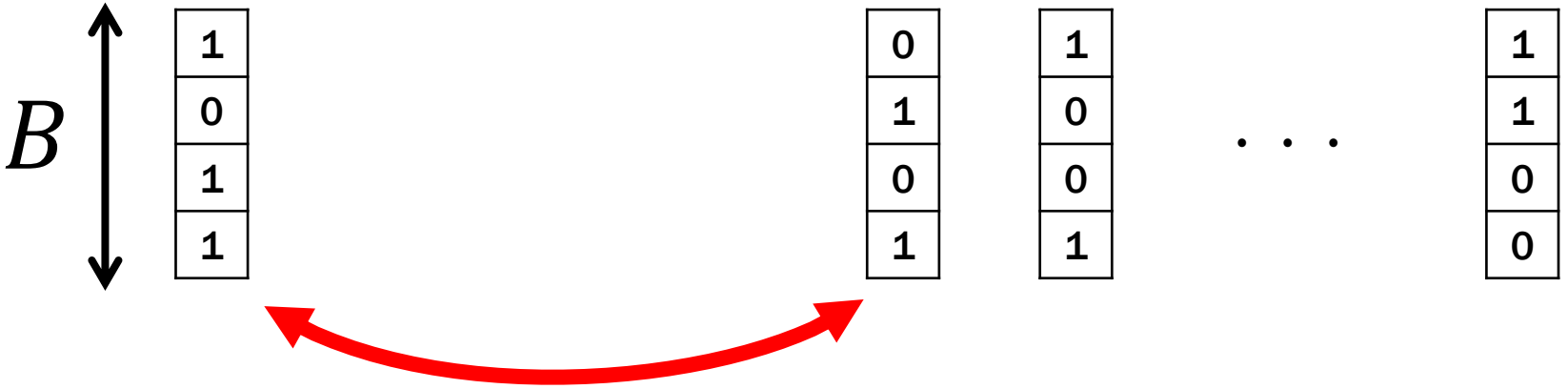
ハミング系手法：メモリ効率



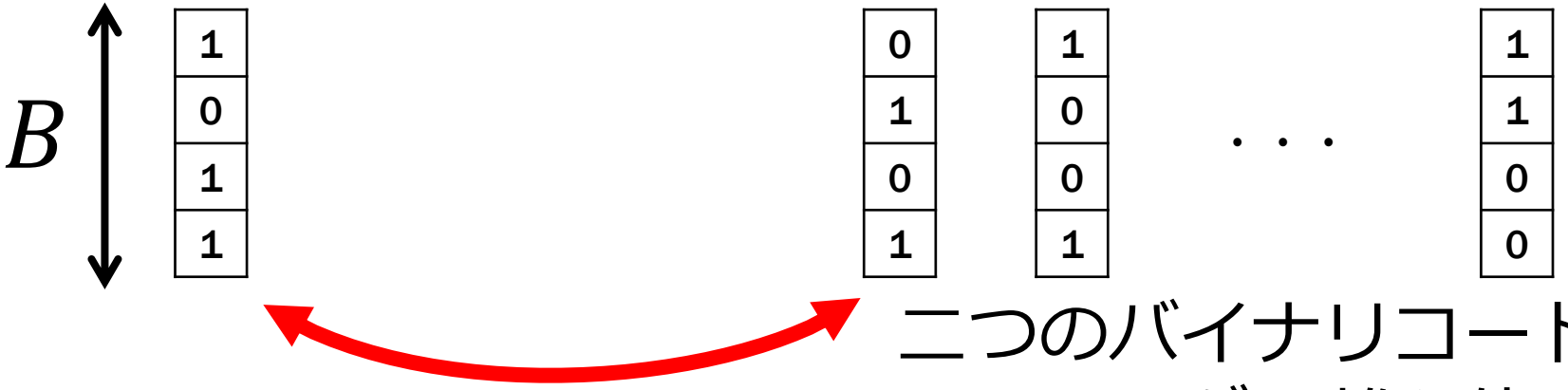
ハミング系手法：メモリ効率



ハミング系手法：距離計算



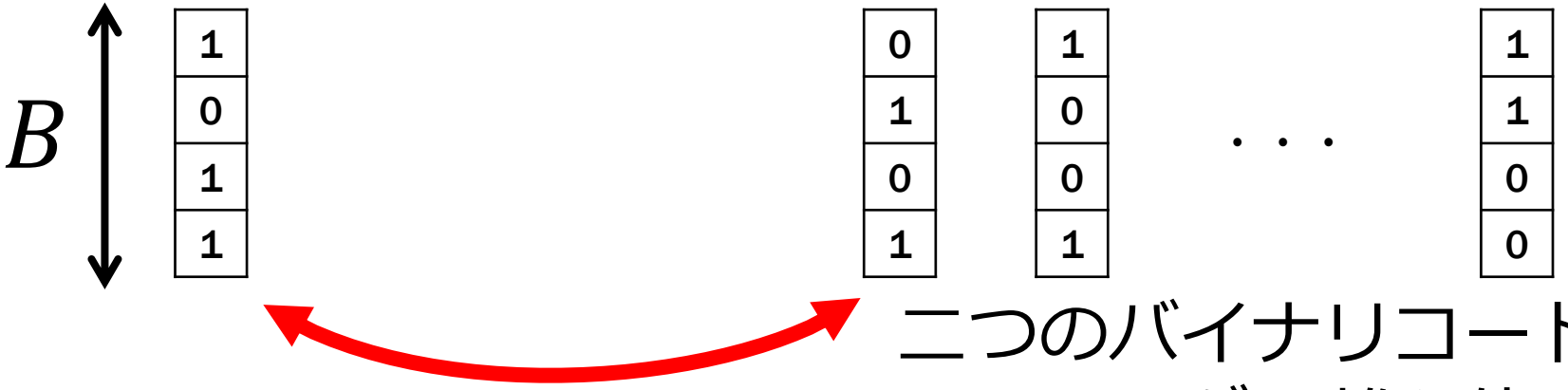
ハミング系手法：距離計算



二つのバイナリコードのハミング距離を使う

$$d_H\left(\begin{matrix}1\\0\\1\\1\end{matrix}, \begin{matrix}0\\1\\0\\1\end{matrix}\right) = 3$$

ハミング系手法：距離計算



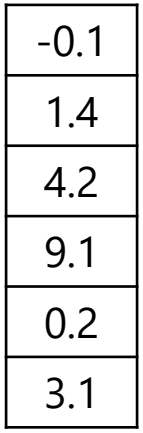
二つのバイナリコードのハミング距離を使う

$$d_H\left(\begin{matrix}1\\0\\1\\1\end{matrix}, \begin{matrix}0\\1\\0\\1\end{matrix}\right) = 3$$

➡ `_mm_popcnt(` $\begin{matrix}1\\0\\1\\1\end{matrix}$ xor $\begin{matrix}0\\1\\0\\1\end{matrix}$ `)`　~10 [ns]

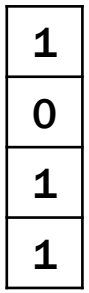
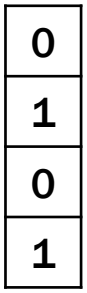
$d_H(\cdot,\cdot)$ は専門の演算命令を使うことで高速に計算できる

ハミング系手法：ハッシュの設計

$d($  ,  $) = 4.61$

Hash

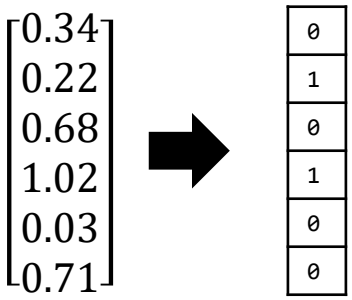
- もし元の距離尺度d が小さければ，ハミング距離d_Hも小さくなる
- そのようなハッシュを設計したい

$d_H($  ,  $) = 3$

ショートコードによる近似最近傍探索

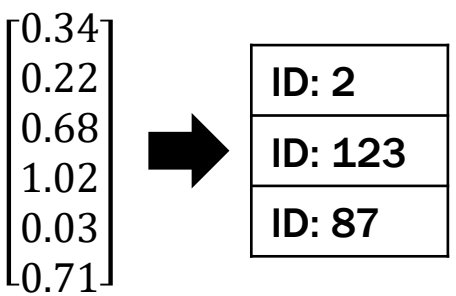
- 基本的な考え方

ハミング系



- 原理
- 手法の例
- 高速計算

ルックアップ系



- 原理
- 探索システム

ハミング系手法の分類

データ非依存：訓練データを用いない

ランダム射影

データ依存：訓練データを用いる

- **教師無し**：ラベル無し訓練データを用いる
  例：訓練画像

Spectral Hashing

- **教師あり**：ラベル付き訓練データを用いる
  例：猫の画像，犬の画像，etc
  ユークリッド距離ではなく，意味を考慮した距離を表現する

- **半教師あり**：ラベル無しおよびラベル付き訓練データを用いる

ハミング系手法の例：ランダム行列

0.23
1.4
-0.3
9.6
0.0
2.3

ハミング系手法の例：ランダム行列

$$\begin{bmatrix} 1.2 & 0.3 & 0.2 & 4.1 & 2.8 & 34.1 \\ -3.1 & 7.5 & -0.2 & 3.4 & 12.3 & -9.6 \\ -5.6 & -4.3 & 12.3 & -5.6 & 7.1 & 5.6 \\ 7.8 & 9.6 & 0.5 & 2.3 & -9.9 & 2.3 \end{bmatrix} \begin{array}{|c|}\hline 0.23 \\ \hline 1.4 \\ \hline -0.3 \\ \hline 9.6 \\ \hline 0.0 \\ \hline 2.3 \\ \hline \end{array}$$

(1) ランダム射影

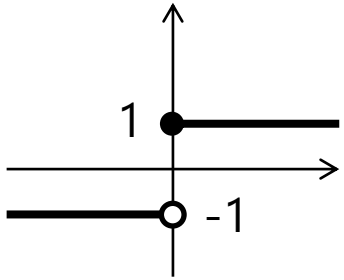
ハミング系手法の例：ランダム行列

$$sgn\left(\begin{bmatrix} 1.2 & 0.3 & 0.2 & 4.1 & 2.8 & 34.1 \\ -3.1 & 7.5 & -0.2 & 3.4 & 12.3 & -9.6 \\ -5.6 & -4.3 & 12.3 & -5.6 & 7.1 & 5.6 \\ 7.8 & 9.6 & 0.5 & 2.3 & -9.9 & 2.3 \end{bmatrix} \begin{bmatrix} 0.23 \\ 1.4 \\ -0.3 \\ 9.6 \\ 0.0 \\ 2.3 \end{bmatrix}\right) = \begin{bmatrix} \mathbf{1} \\ \mathbf{-1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}$$

(1) ランダム射影

(2) 閾値処理

$sgn(\cdot)$:



注：バイナリコードの値として{1, -1}を採用するか{1, 0}を採用するかは任意（変換可能）

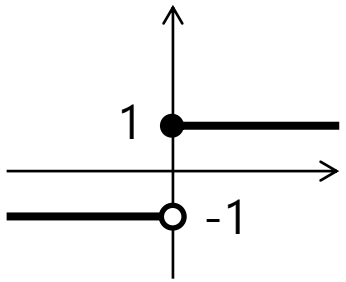
ハミング系手法の例：ランダム行列

$$sgn\left(\begin{bmatrix} 1.2 & 0.3 & 0.2 & 4.1 & 2.8 & 34.1 \\ -3.1 & 7.5 & -0.2 & 3.4 & 12.3 & -9.6 \\ -5.6 & -4.3 & 12.3 & -5.6 & 7.1 & 5.6 \\ 7.8 & 9.6 & 0.5 & 2.3 & -9.9 & 2.3 \end{bmatrix} \begin{bmatrix} 0.23 \\ 1.4 \\ -0.3 \\ 9.6 \\ 0.0 \\ 2.3 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

(1) ランダム射影

(2) 閾値処理

$sgn(\cdot)$:



注：バイナリコードの値として{1, -1}を採用するか{1, 0}を採用するかは任意（変換可能）

☺シンプル
☹精度が低い ➡ 「データの分布」を利用しよう

ハミング系手法の分類

データ非依存：訓練データを用いない

ランダム射影

データ依存：訓練データを用いる

- **教師無し**：ラベル無し訓練データを用いる
  例：訓練画像

Spectral Hashing

- **教師あり**：ラベル付き訓練データを用いる
  例：猫の画像，犬の画像，etc
  ユークリッド距離ではなく，意味を考慮した距離を表現する

- **半教師あり**：ラベル無しおよびラベル付き訓練データを用いる

ハミング系手法の例：Spectral Hashing

原則：
- 元の距離尺度d が小さければ，ハミング距離d_Hも小さくなる
- そのような「変換」を設計したい

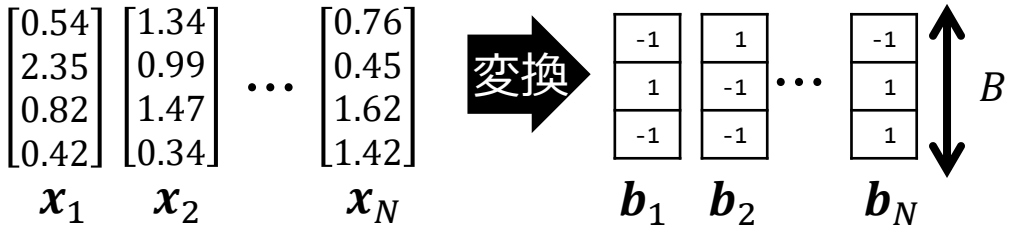
＋

無駄のないコードを作成したいという意図から条件を追加：
- 変換後の各ビットがバランスしている
  - 各ビットは{-1, +1}を半々ずつもつ
- 各ビットには相関が無い

Y. Weiss et al., “Spectral Hashing”, NIPS 2008.

既知：元のベクトル

未知：ショートコード



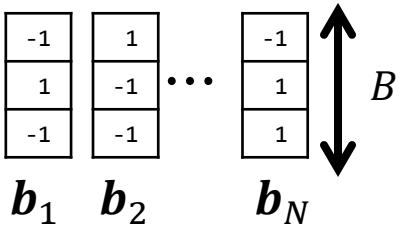
既知：元のベクトル

$$\begin{bmatrix}0.54\\2.35\\0.82\\0.42\end{bmatrix}\ \begin{bmatrix}1.34\\0.99\\1.47\\0.34\end{bmatrix}\ \cdots\ \begin{bmatrix}0.76\\0.45\\1.62\\1.42\end{bmatrix}$$

$\boldsymbol{x}_1$ $\boldsymbol{x}_2$ $\boldsymbol{x}_N$

変換

未知：ショートコード

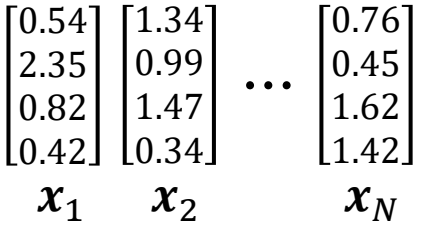


$\boldsymbol{b}_1$ $\boldsymbol{b}_2$ $\boldsymbol{b}_N$

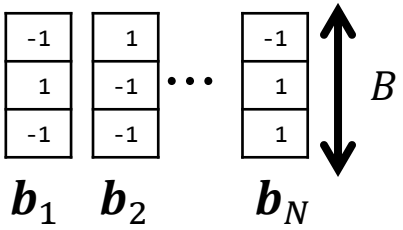
元のベクトル間の類似度行列：

$$W \in \mathbb{R}^{N\times N},\ W_{ij} = e^{\frac{-\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|^2}{\epsilon^2}}$$

既知：元のベクトル



未知：ショートコード



元のベクトル間の類似度行列：

$$W \in \mathbb{R}^{N\times N},\ W_{ij} = e^{\frac{-\|x_i - x_j\|^2}{\epsilon^2}}$$

次の最適化問題を解くことで，未知のショートコードを求める

$$\min_{\boldsymbol{b}_1,\boldsymbol{b}_2,\ldots,\boldsymbol{b}_N} \sum_{i,j} W_{ij} \|\boldsymbol{b}_i - \boldsymbol{b}_j\|^2$$

$\boldsymbol{b}_i$と$\boldsymbol{b}_j$を近くする（と、この最適化問題の良い解になる）

$\boldsymbol{x}_i$と$\boldsymbol{x}_j$の類似度が大きいときは・・

s.t.

$$\sum_i \boldsymbol{b}_i = \mathbf{0}$$

ビットは
バランス

$$\frac{1}{N}\sum_i \boldsymbol{b}_i \boldsymbol{b}_i^T = I$$

ビットは
無相関

$$\boldsymbol{b}_i \in \{-1, 1\}^B, \forall i$$

ビットは
バイナリ

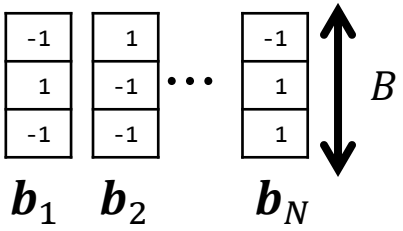
既知：元のベクトル

$$\begin{bmatrix}0.54\\2.35\\0.82\\0.42\end{bmatrix}\ \begin{bmatrix}1.34\\0.99\\1.47\\0.34\end{bmatrix}\ \cdots\ \begin{bmatrix}0.76\\0.45\\1.62\\1.42\end{bmatrix}$$

$\boldsymbol{x}_1$ $\boldsymbol{x}_2$ $\boldsymbol{x}_N$

変換

未知：ショートコード



$\boldsymbol{b}_1$ $\boldsymbol{b}_2$ $\boldsymbol{b}_N$

元のベクトル間の類似度行列：

$$W \in \mathbb{R}^{N\times N},\ W_{ij} = e^{\frac{-\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|^2}{\epsilon^2}}$$

次の最適化問題を解くことで，未知のショートコードを求める

$$\min \sum W_{ij}\|\boldsymbol{b}_i - \boldsymbol{b}_j\|^2$$

ビットはバイナリであるという条件を取り除く（スペクトラル緩和）と，グラフラプラシアンの一般化固有値問題として解ける [Weiss+, NIPS08]

$$\sum_i \boldsymbol{b}_i = \mathbf{0} \qquad \frac{1}{N}\sum_i \boldsymbol{b}_i\boldsymbol{b}_i^T = I \qquad \boldsymbol{b}_i \in \{-1,1\}^B, \forall i$$

ビットは
バランス

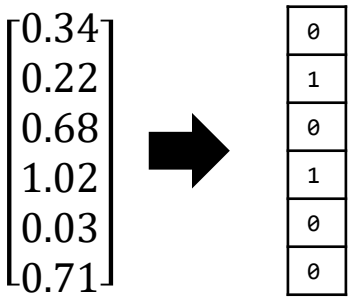
ビットは
無相関

ビットは
バイナリ

ショートコードによる近似最近傍探索

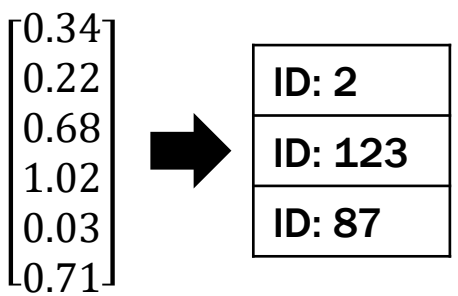
- 基本的な考え方

ハミング系



- 原理
- 手法の例
- 高速計算

ルックアップ系



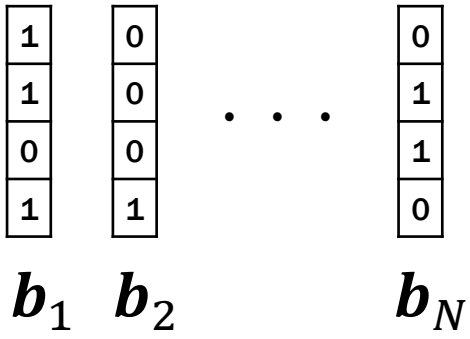
- 原理
- 探索システム

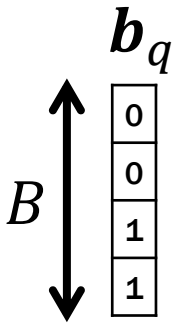
ハミング系手法の高速計算

- ハミング距離による線形探索は高速だが，計算量は$O(BN)$
- 依然Nに対し線形であり，Nが大きいと遅い

- **ハッシュテーブル**を用いて，全く同様の結果を高速に計算出来る

ハミング系手法の高速計算：データ登録

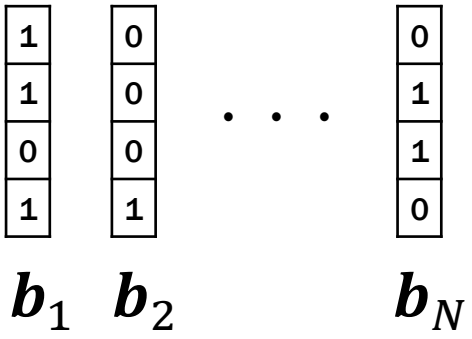


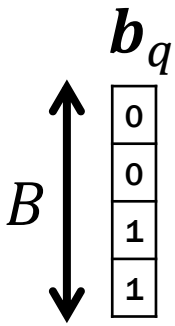



```
uint4 query
```

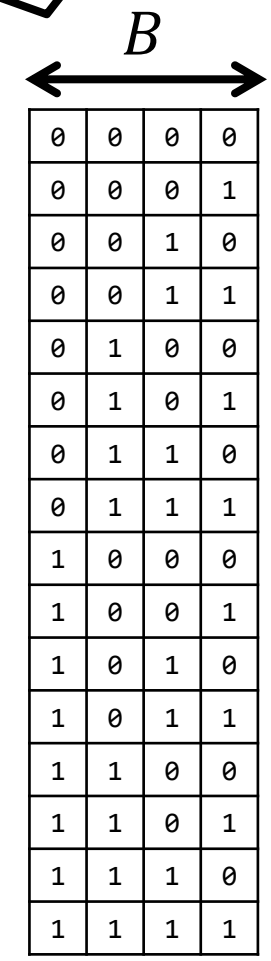
ハミング系手法の高速計算：データ登録

[0000]から[1111]までの値を持つ
エントリを用意しておく



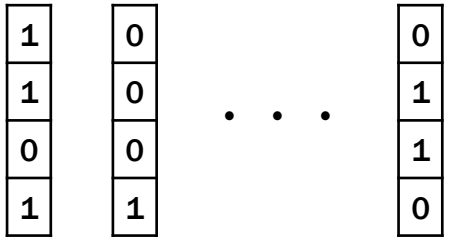


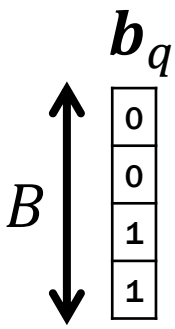
uint4 query



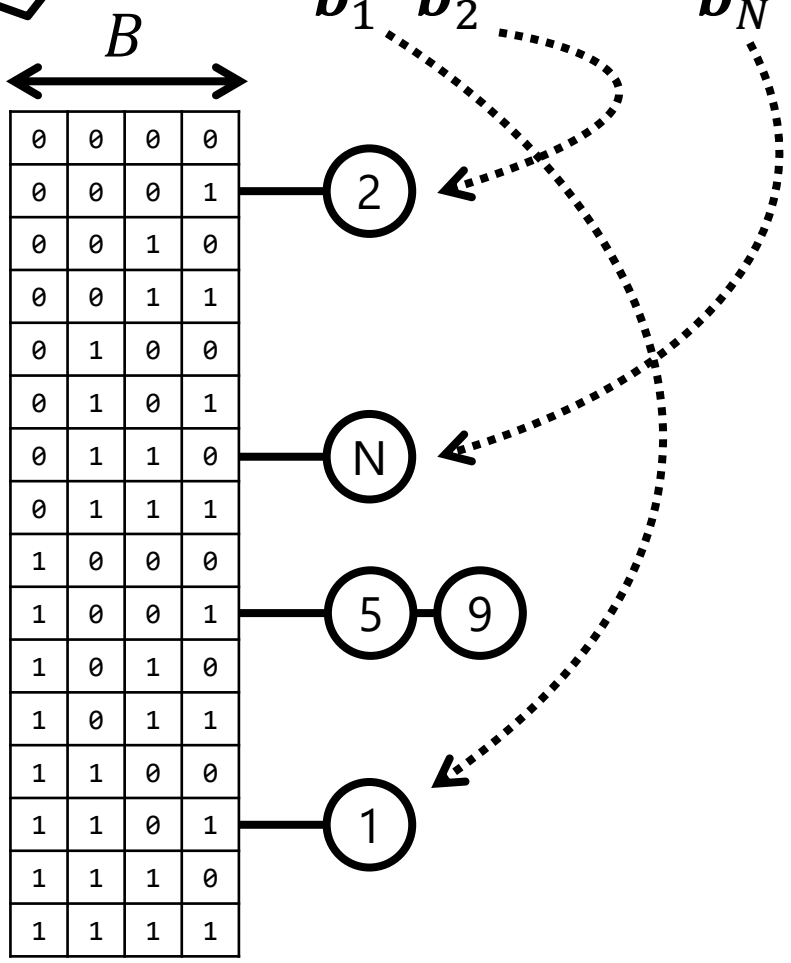
ハミング系手法の高速計算：データ登録

[0000]から[1111]までの値を持つ
エントリを用意しておく





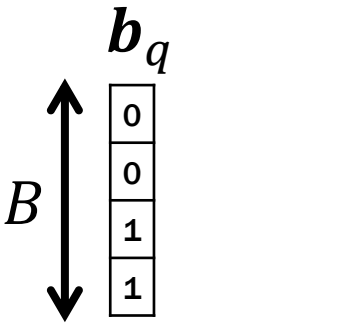
uint4 query



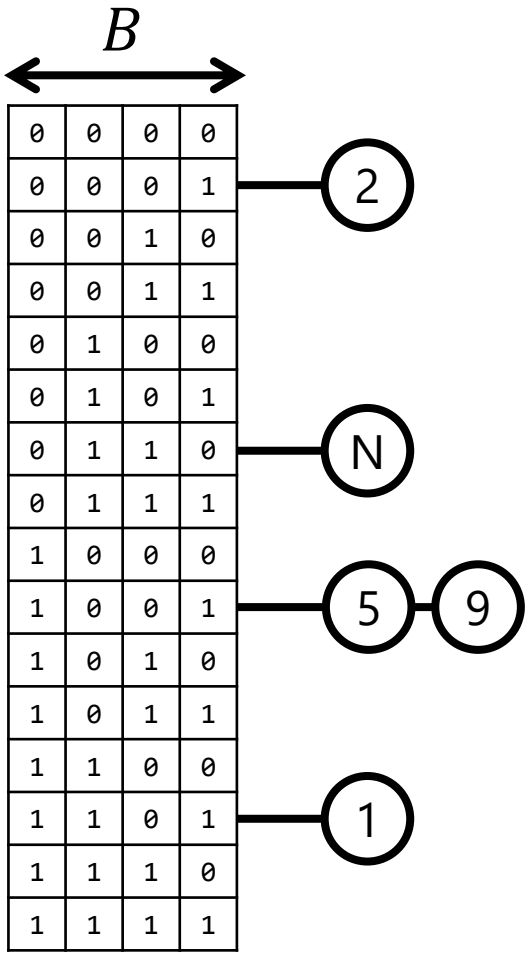
各コードについて，それ
自身をエントリとして，
番号を挿入

ハミング系手法の高速計算：探索

vec<list<int>> table

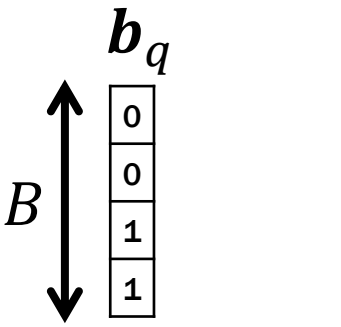


uint4 query

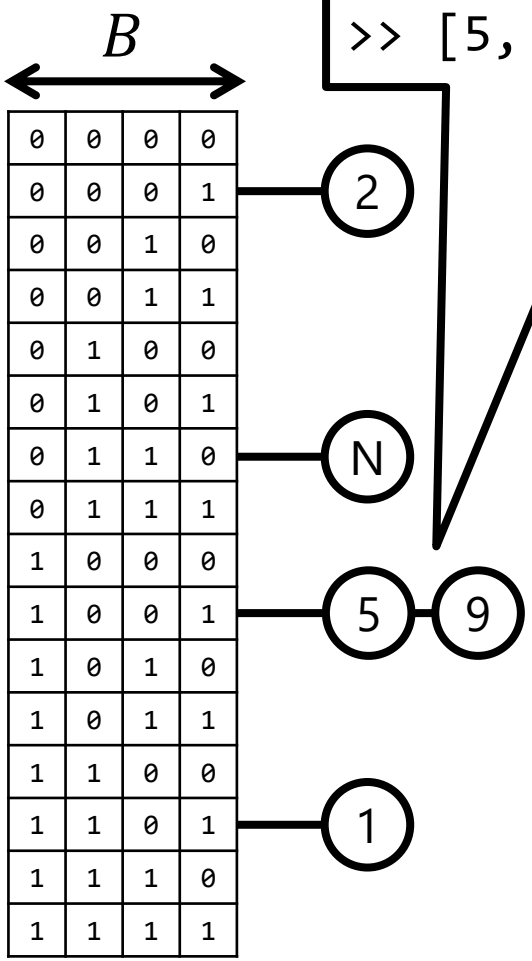


ハミング系手法の高速計算：探索

vec<list<int>> table



uint4 query



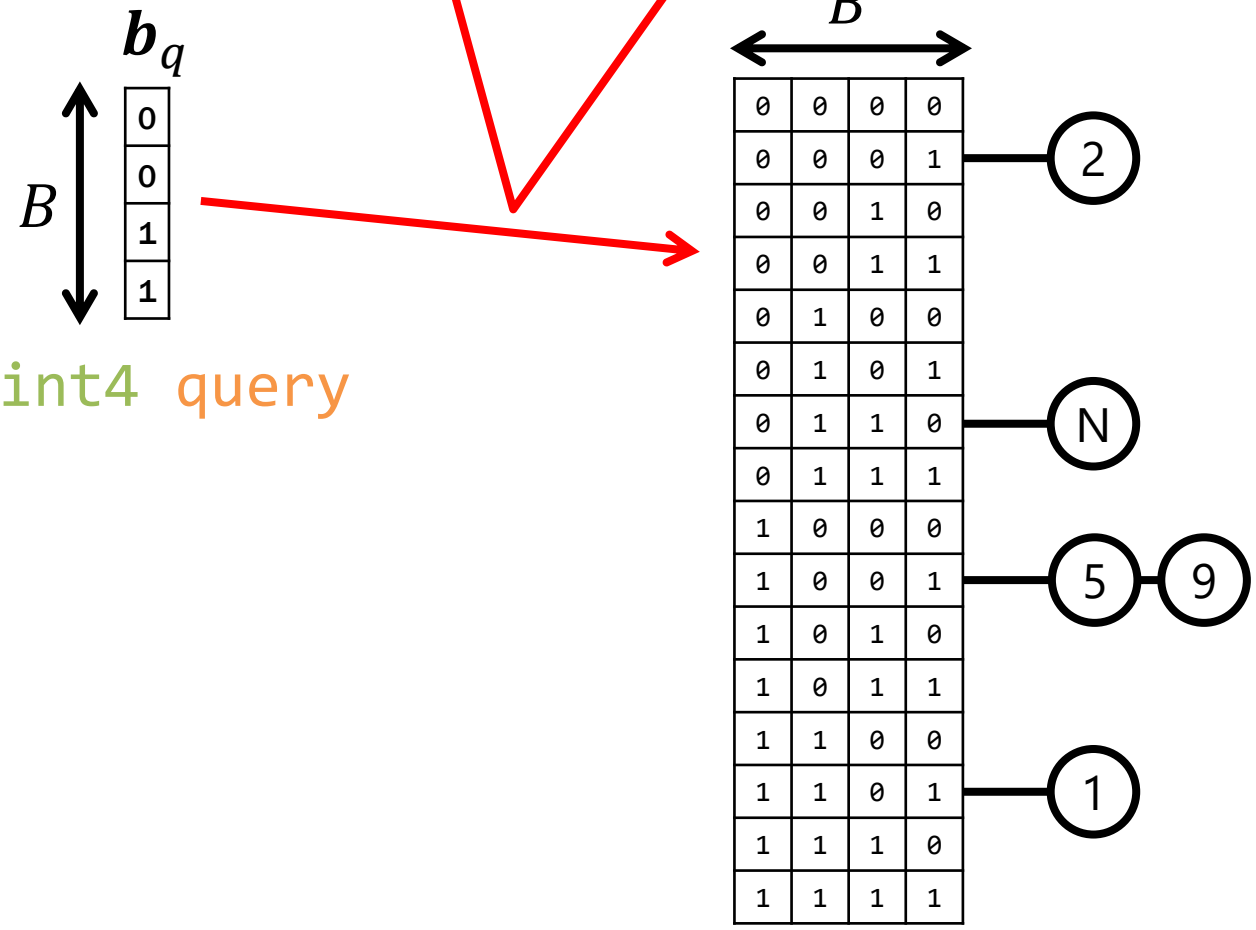
例：

```
list<int> v = table[9];
Print(v);
>> [5, 9]
```

ハミング系手法の高速計算：探索

vec<list<int>> table

(1) クエリと同じコードがあるか？
配列アクセスなので，$O(1)$.
すなわち，table[query]

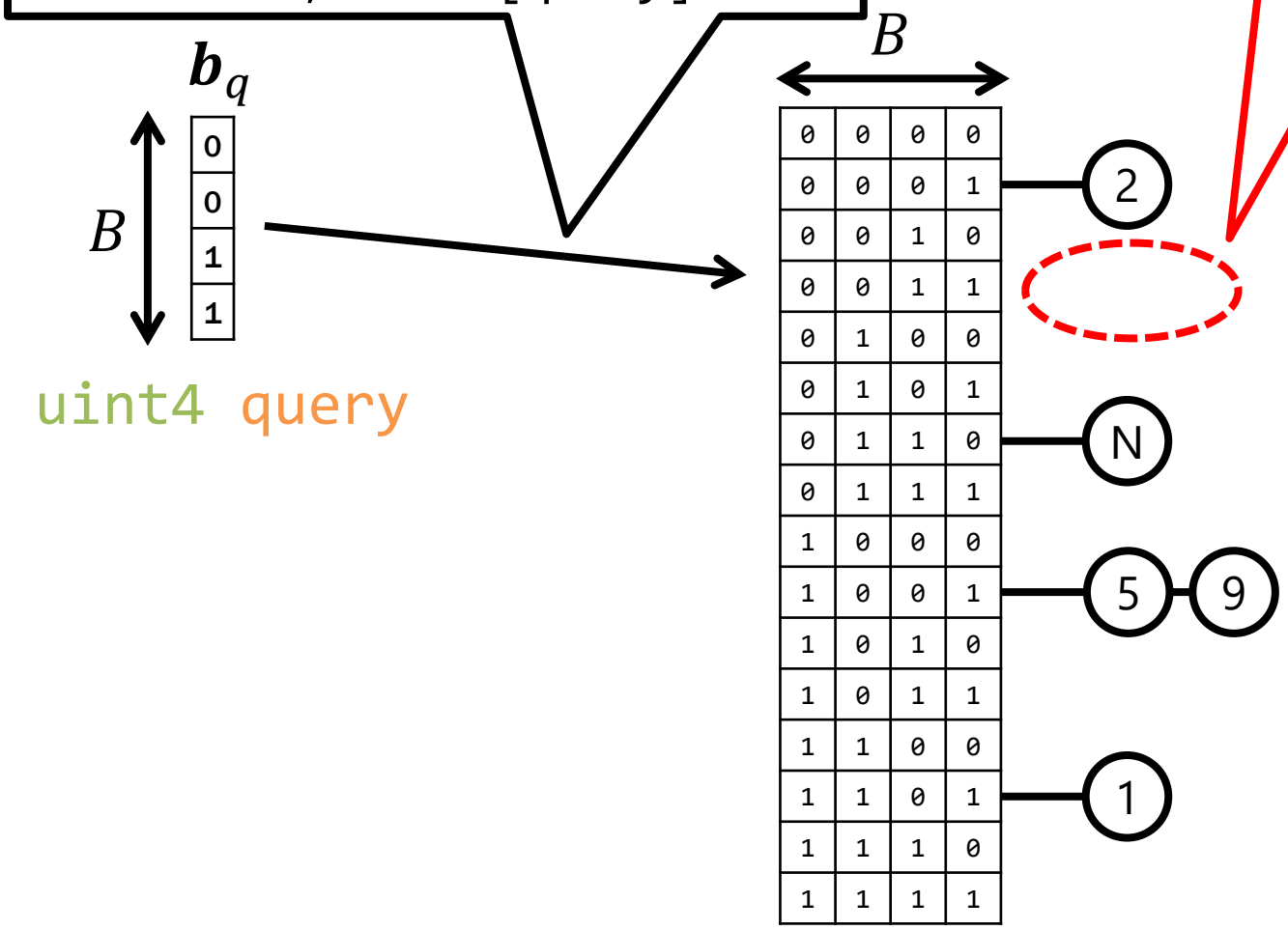


ハミング系手法の高速計算：探索

vec<list<int>> table

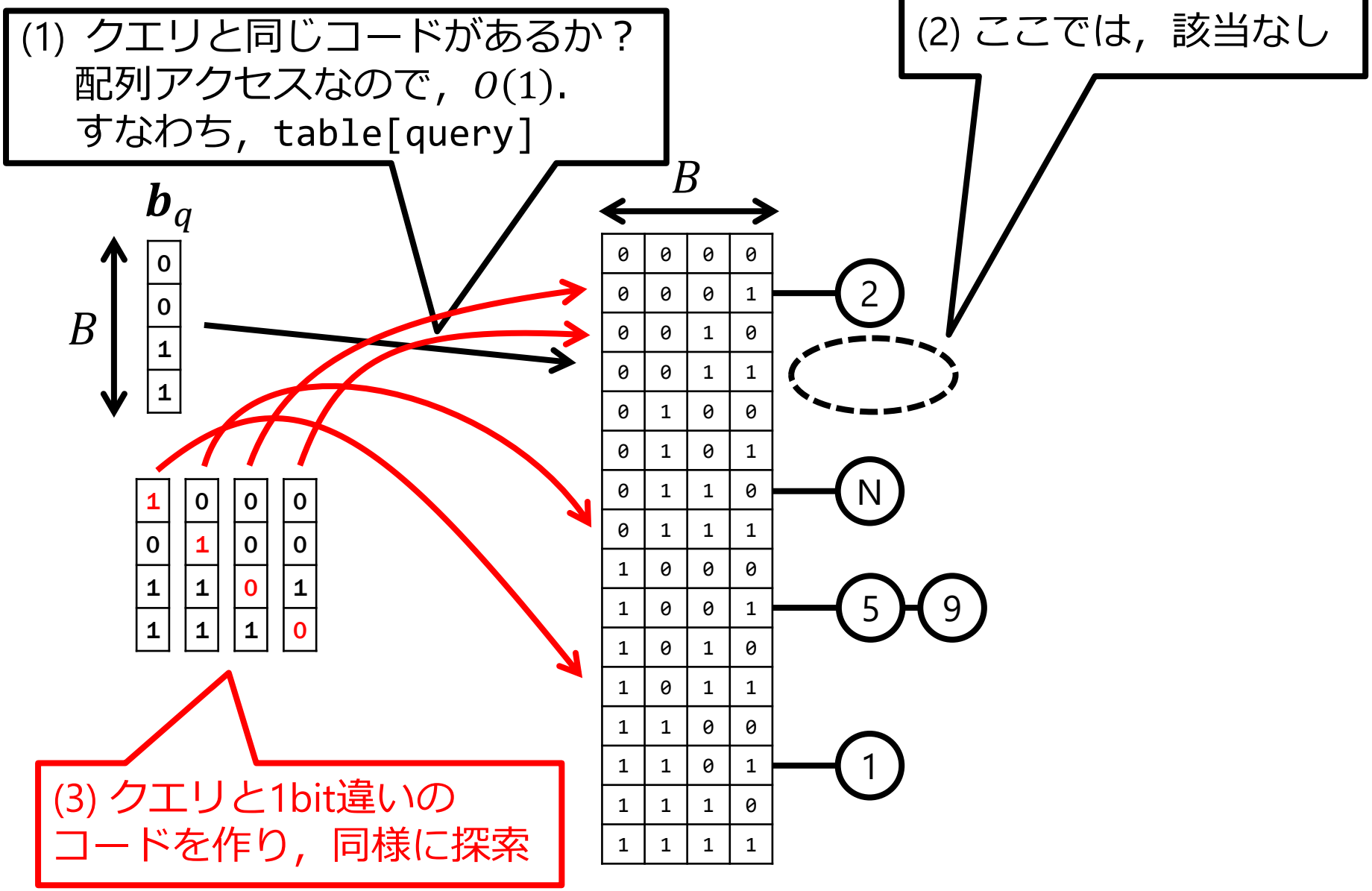
(1) クエリと同じコードがあるか？
配列アクセスなので，$O(1)$.
すなわち，table[query]

(2) ここでは，該当なし



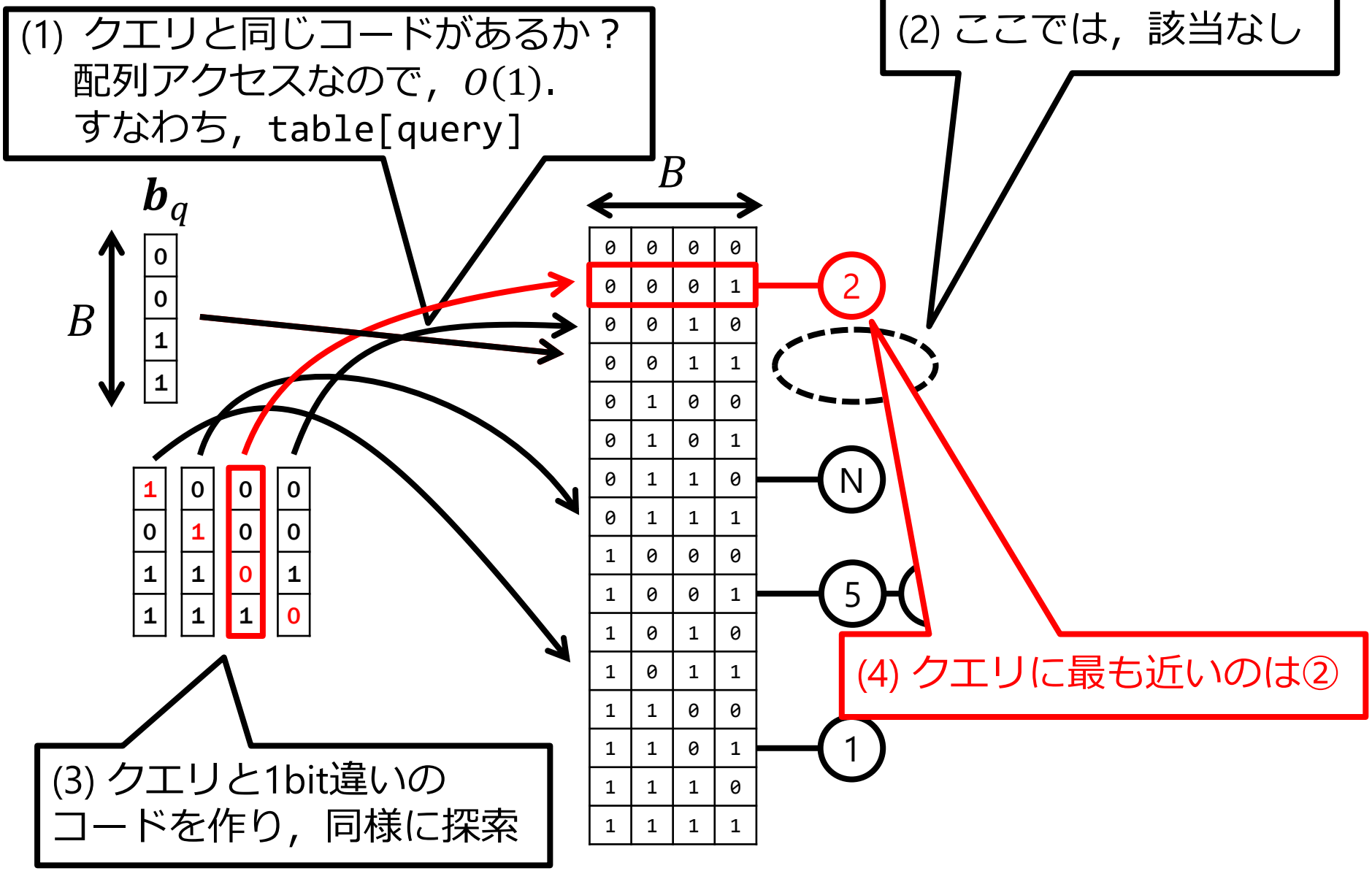
ハミング系手法の高速計算：探索

vec<list<int>> table

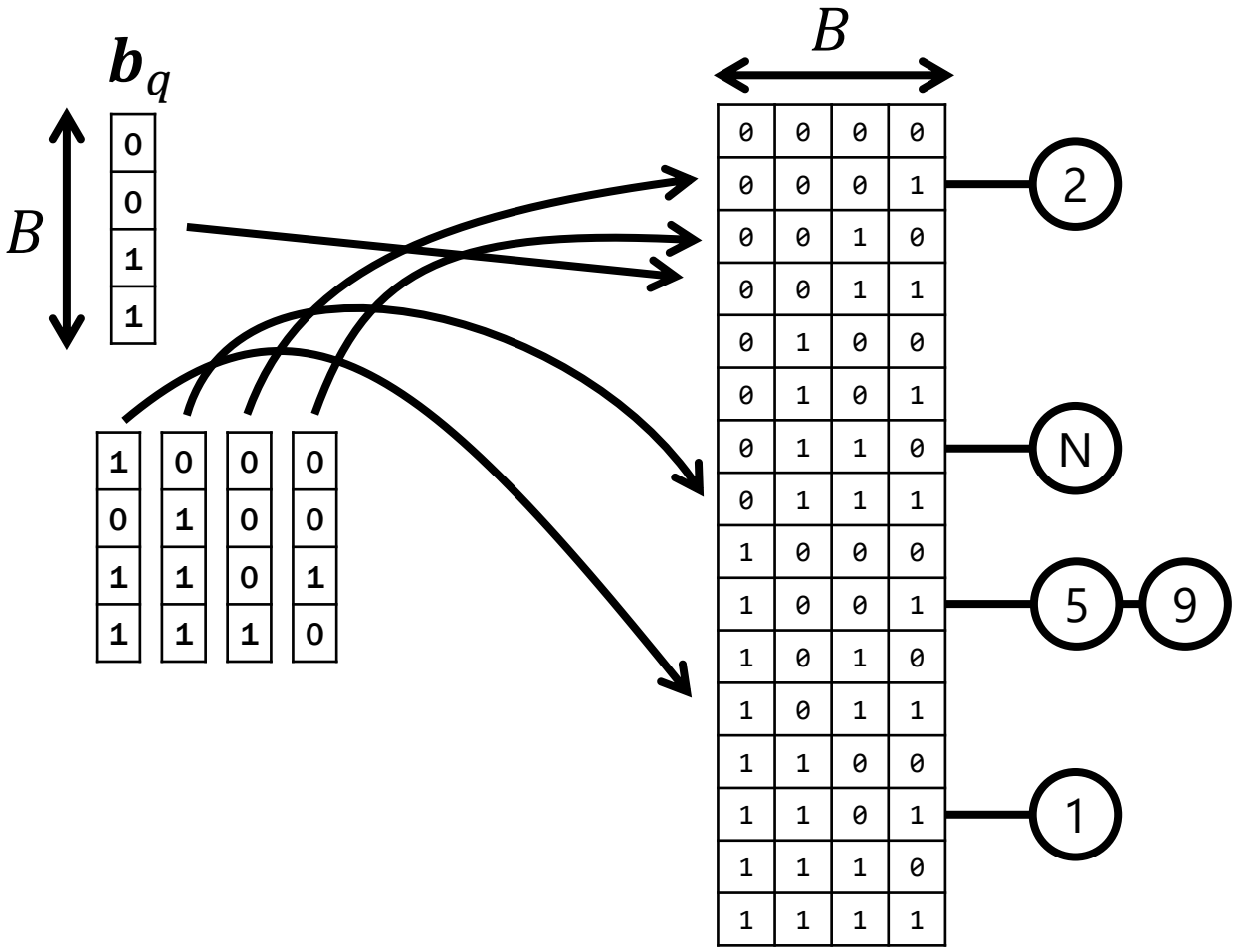


ハミング系手法の高速計算：探索

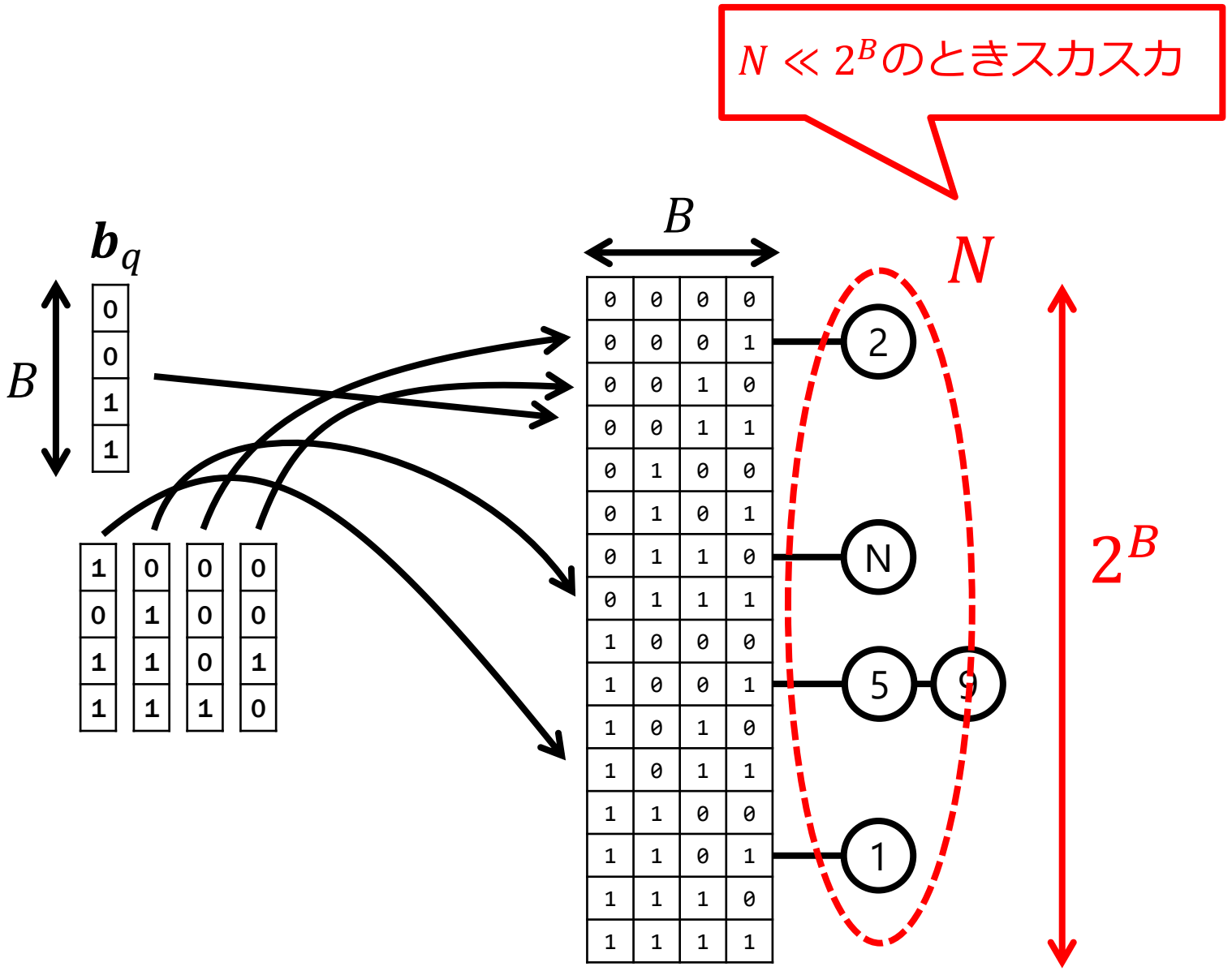
vec<list<int>> table



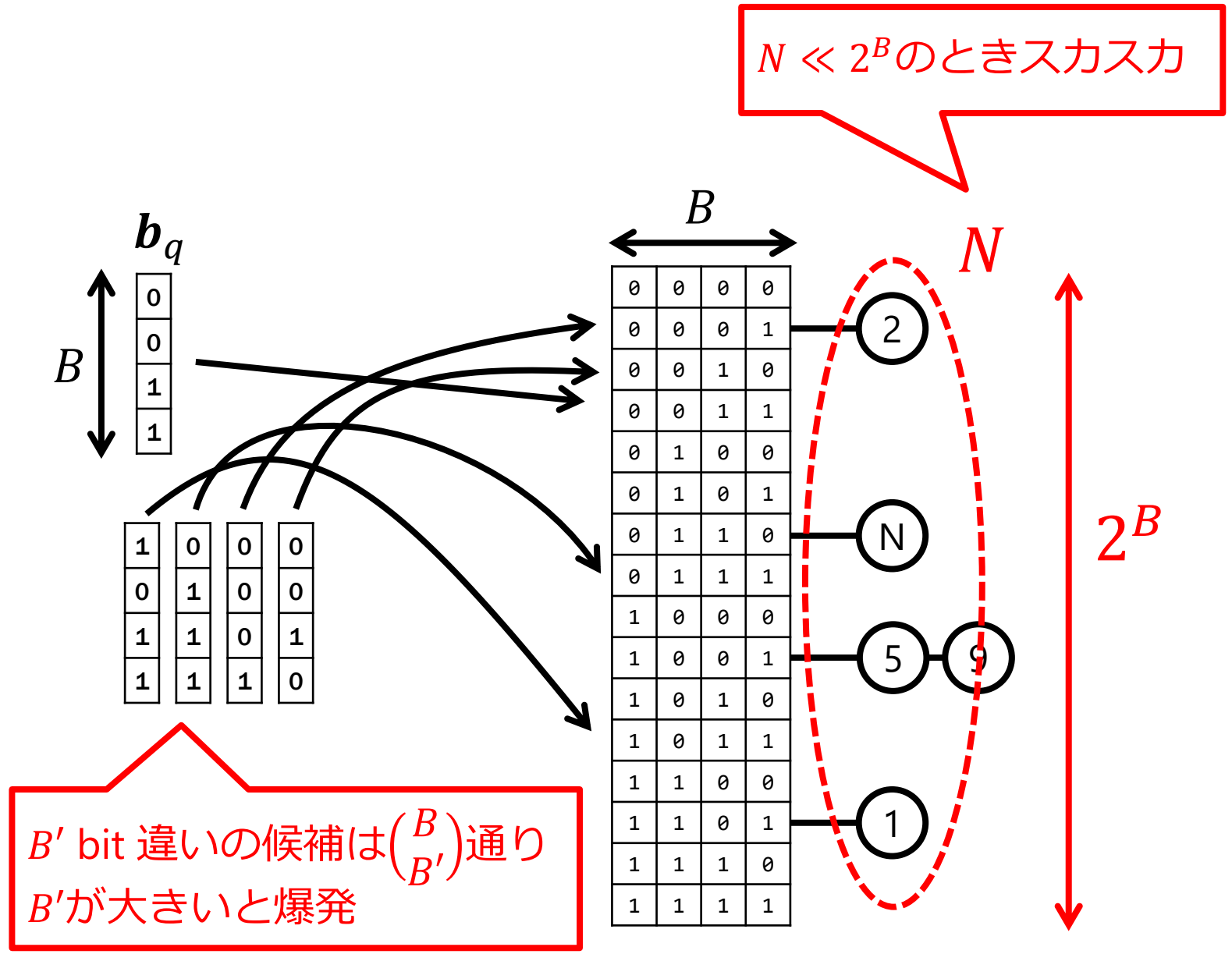
ハミング系手法の高速計算：問題と発展



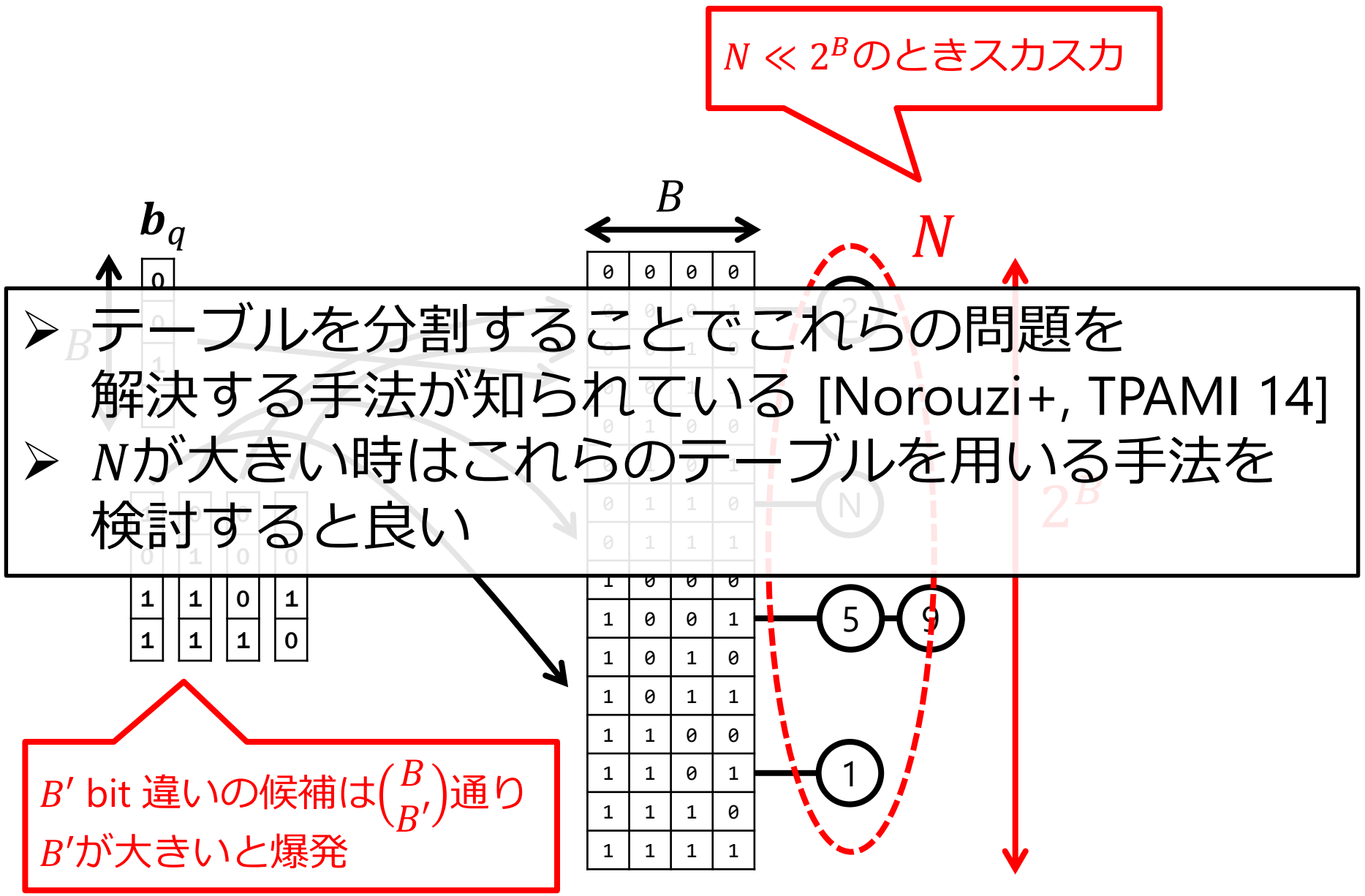
ハミング系手法の高速計算：問題と発展



ハミング系手法の高速計算：問題と発展



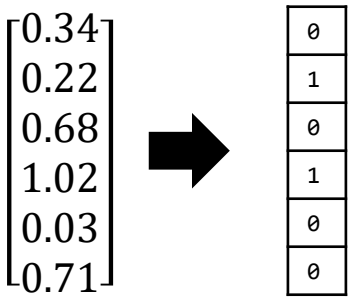
ハミング系手法の高速計算：問題と発展



ショートコードによる近似最近傍探索

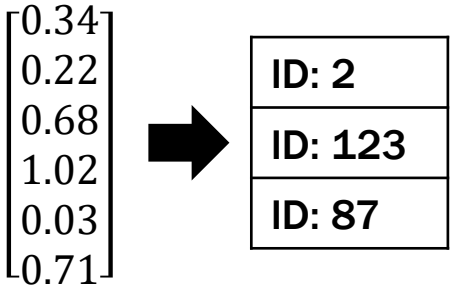
- 基本的な考え方

ハミング系



- 原理
- 手法の例
- 高速計算

ルックアップ系

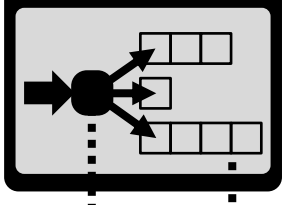


- 原理
- 探索システム

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った
探索システム



事前変換

Cartesian k-means
[Norouzi CVPR 13]

同じ

Optimized PQ
[Ge, CVPR13][Ge, TPAMI 14]

各次元を複数
コードブックで表す

Optimized Cartesian k-means [Wang, TKDE 14]

Tree quantization
[Babenko, CVPR 15]

PQそのものの発展

一般化

Additive quantization
[Babenko, CVPR 14]

立式が同じ

Composite quantization
[Zhang, ICML 14]

新しい問題設定

高速符号化 [Zhang, CVPR 15]

教師付き [Wang, CVPR16]

マルチモーダル [Zhang, CVPR16]

疎量子化：
k-means

リランキング：
残差PQコード

疎量子化の工夫

複数k-means [Xia, ICCV 13]

PQ [Babenko, CVPR 12]

ジャーナル版

OPQ+local codebook
[Babenko, TPAMI 15]

列挙の高速化 [Iwamura, ICCV 13]

残差量子化 [Babenko, CVPR 16]

リランキングの工夫

二段階 [Jégou, ICASSP 11]

local codebook [Kalantidis, CVPR 14]

リランキング側残差考慮 [Heo, CVPR 16]

PQを使った
探索システムの発展

その他トピック

余分ビット [Heo, CVPR 14]

ハッシュテーブル
[Matsui, ICCV 15]

GPU
[Wieschollek, CVPR 16]

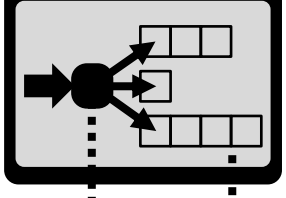
キャッシュ
[André, VLDB 15]

画像検索システム
[Jégou, PAMI 12]
[Spyromitros-Xioufis, TMM 14]

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った探索システム



事前変換

Cartesian k-means [Norouzi CVPR 13]

同じ

Optimized PQ [Ge, CVPR13][Ge, TPAMI 14]

各次元を複数コードブックで表す

Optimized Cartesian k-means [Wang, TKDE 14]

Tree quantization [Babenko, CVPR 15]

PQそのものの発展

一般化

Additive quantization [Babenko, CVPR 14]

立式が同じ

Composite quantization [Zhang, ICML 14]

新しい問題設定

高速符号化 [Zhang, CVPR 15]

教師付き [Wang, CVPR16]

マルチモーダル [Zhang, CVPR16]

疎量子化：k-means

リランキング：残差PQコード

疎量子化の工夫

複数k-means [Xia, ICCV 13]

PQ [Babenko, CVPR 12]

ジャーナル版

OPQ+local codebook [Babenko, TPAMI 15]

列挙の高速化 [Iwamura, ICCV 13]

残差量子化 [Babenko, CVPR 16]

リランキングの工夫

二段階 [Jégou, ICASSP 11]

local codebook [Kalantidis, CVPR 14]

リランキング側残差考慮 [Heo, CVPR 16]

PQを使った探索システムの発展

その他トピック

余分ビット [Heo, CVPR 14]

ハッシュテーブル [Matsui, ICCV 15]

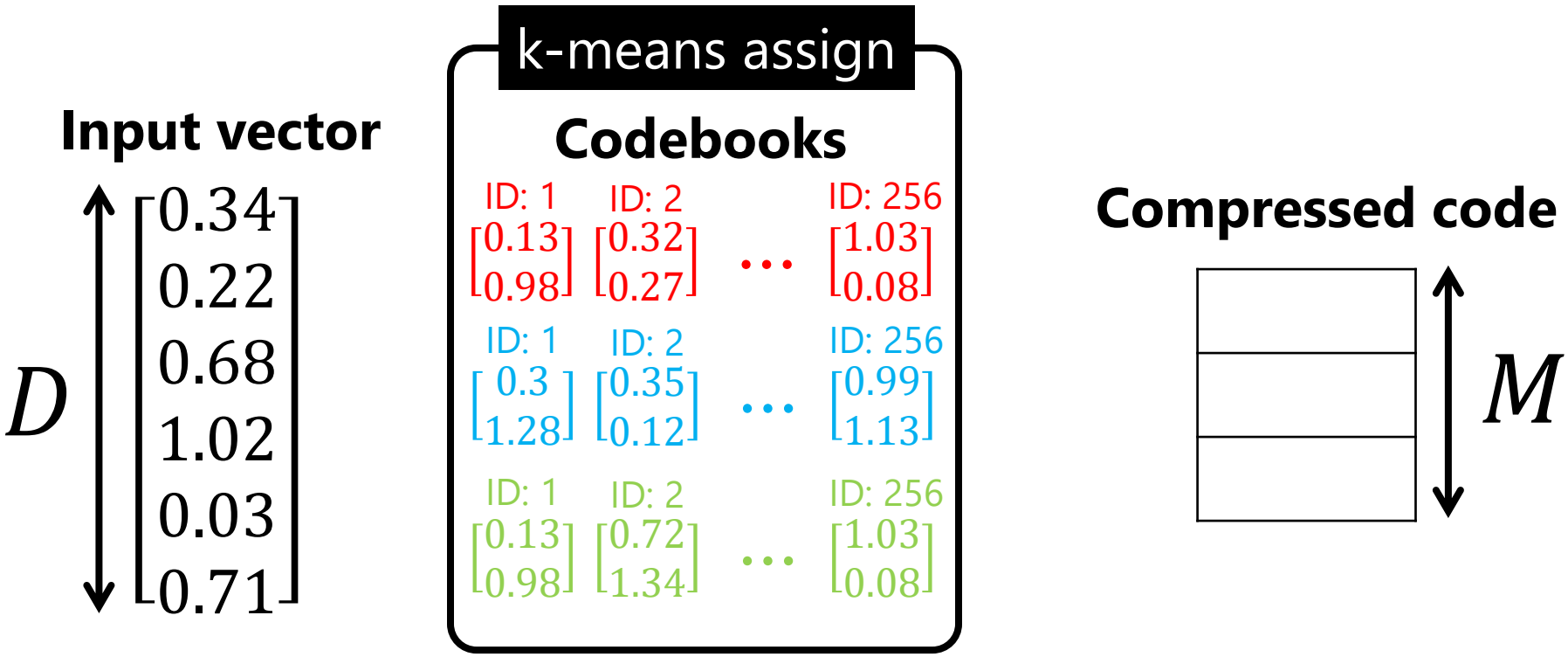
GPU [Wieschollek, CVPR 16]

キャッシュ [André, VLDB 15]

画像検索システム [Jégou, PAMI 12] [Spyromitros-Xioufis, TMM 14]

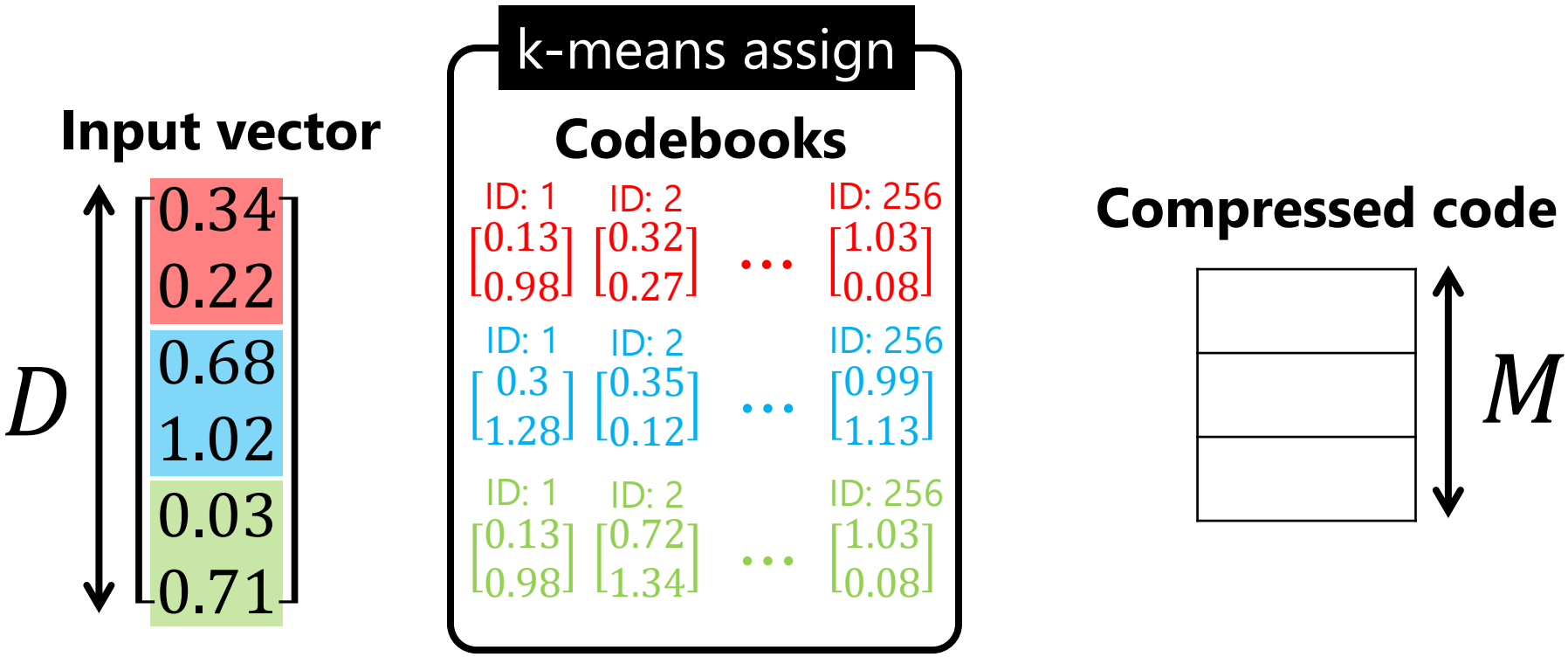
Product Quantization [Jégou, TPAMI 2011]

➢ ベクトルを分割してk-meansする



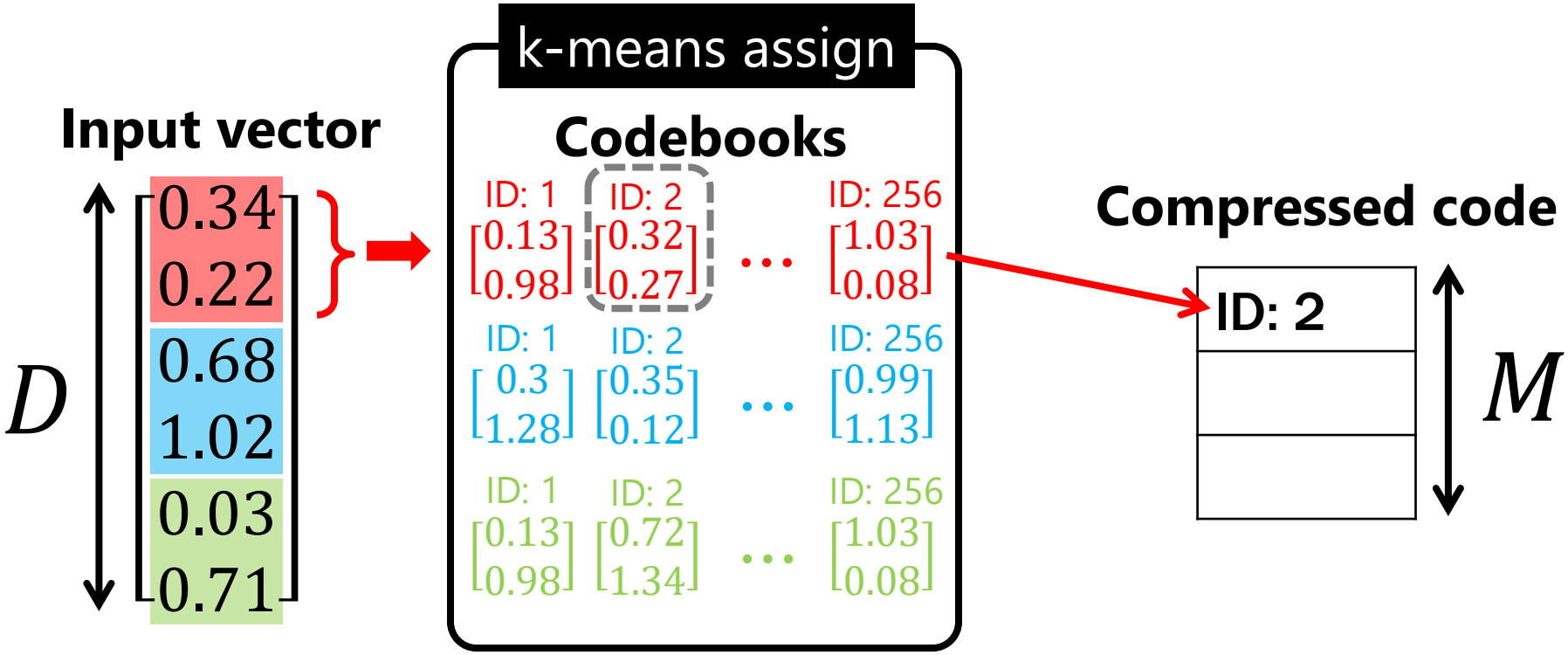
Product Quantization [Jégou, TPAMI 2011]

➢ ベクトルを分割してk-meansする



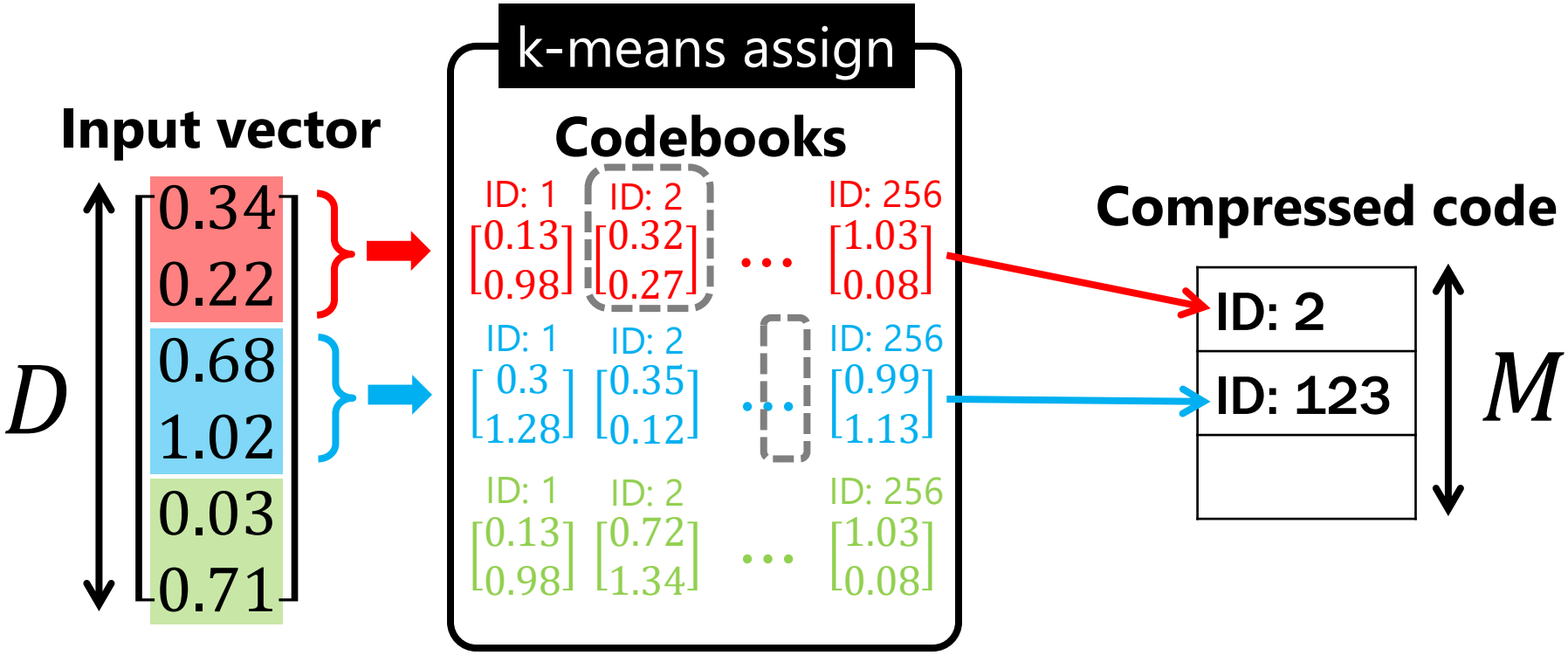
Product Quantization [Jégou, TPAMI 2011]

➢ ベクトルを分割してk-meansする



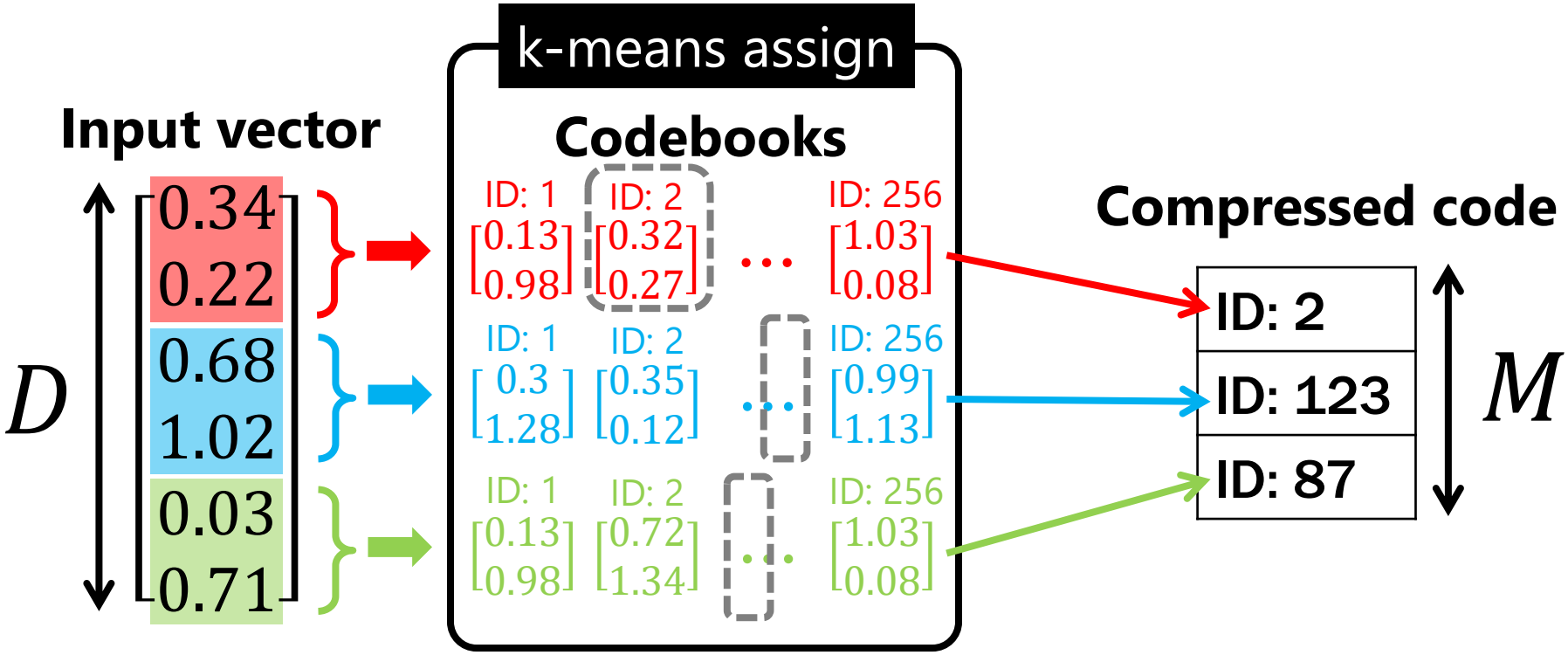
Product Quantization [Jégou, TPAMI 2011]

➢ ベクトルを分割してk-meansする



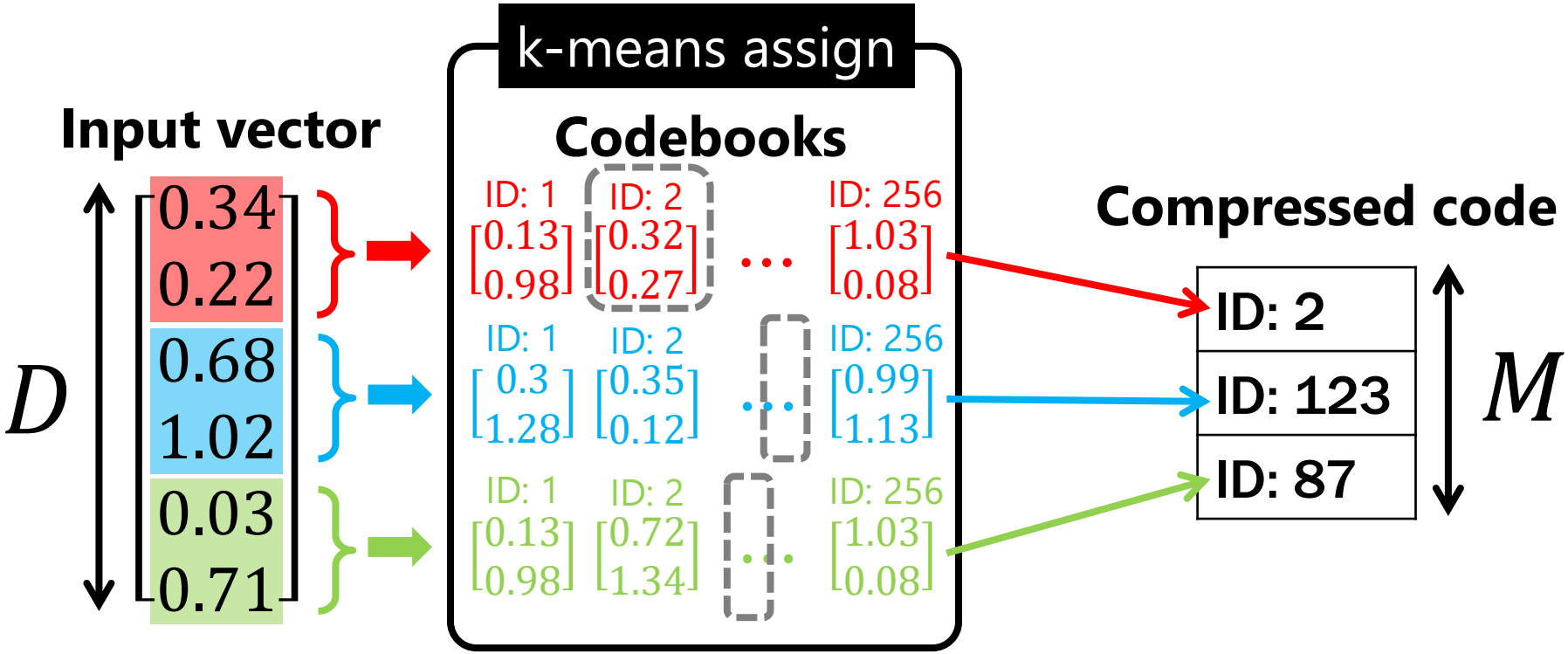
Product Quantization [Jégou, TPAMI 2011]

➢ ベクトルを分割してk-meansする



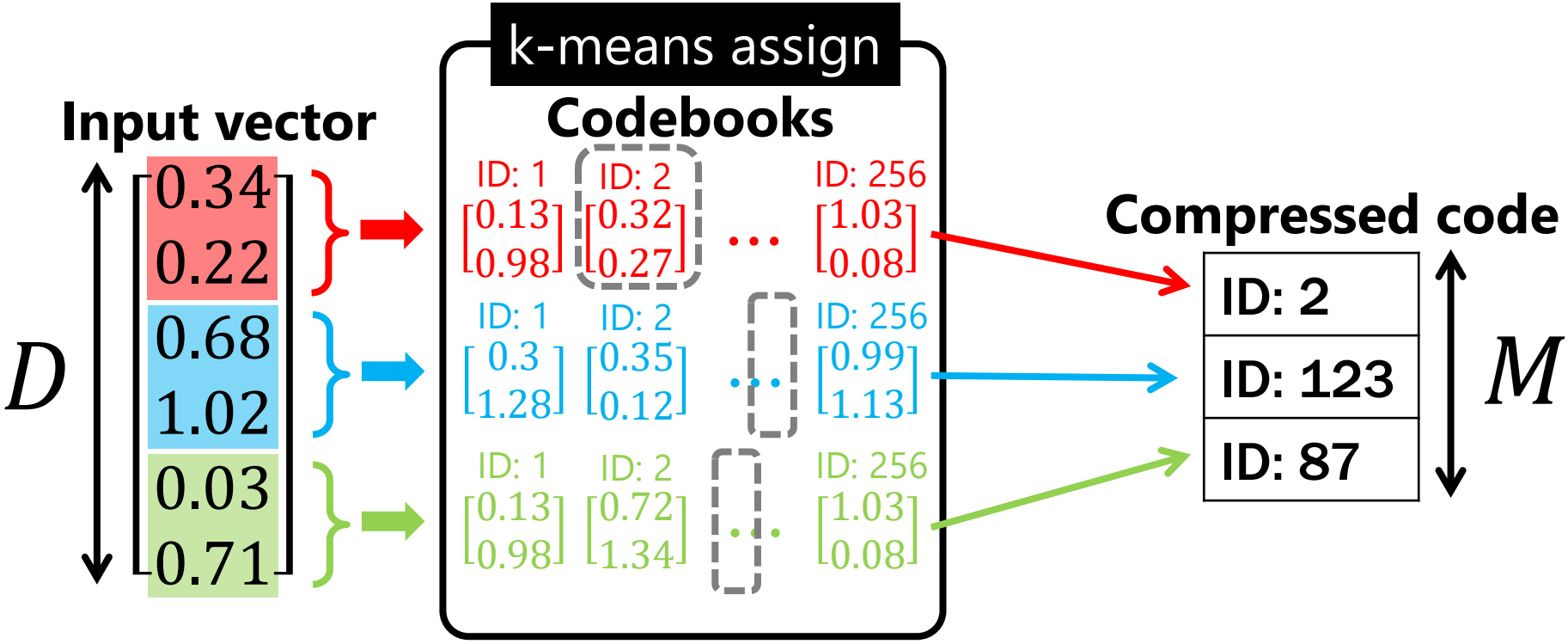
Product Quantization [Jégou, TPAMI 2011]

- ベクトルを分割してk-meansする

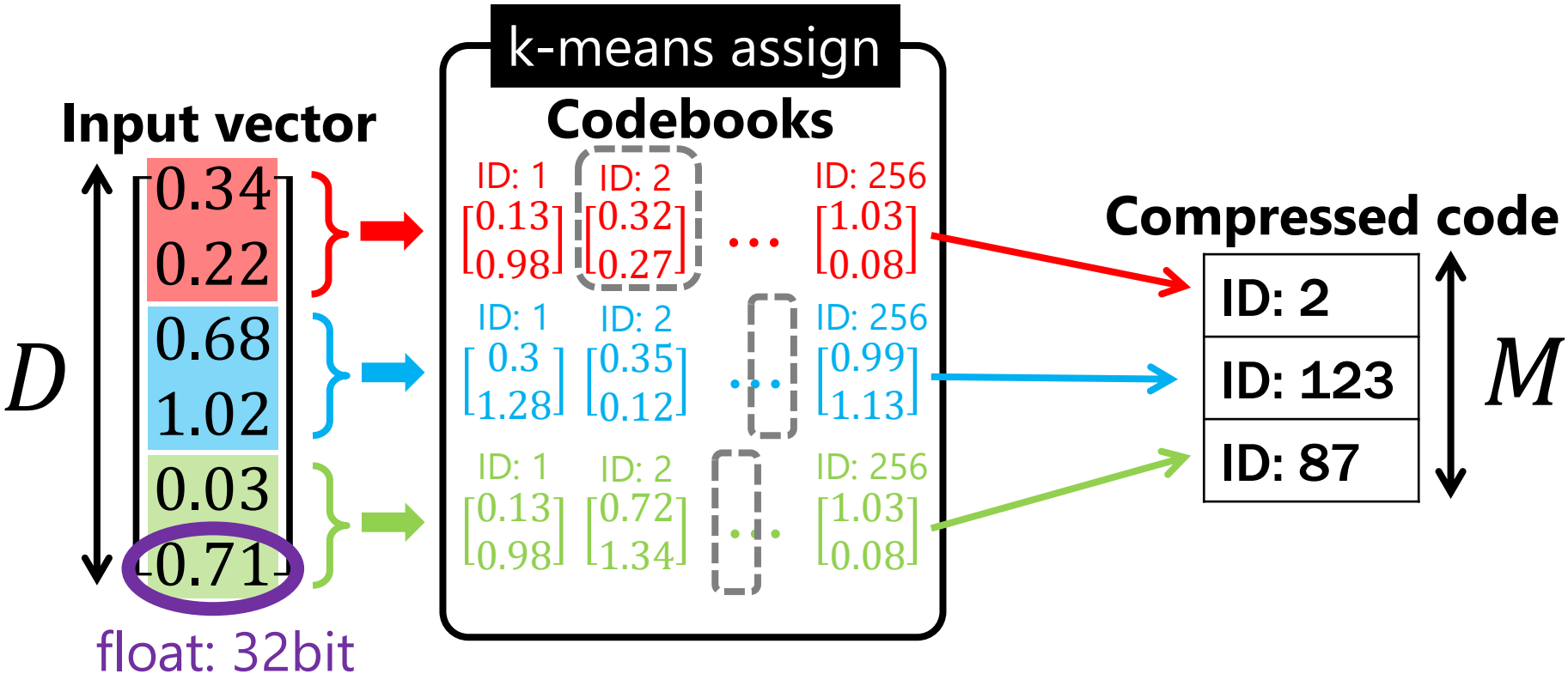


- 単純
- メモリ効率良い
- 距離 $d(input, code)^2$ を近似計算可能

Product Quantization: メモリ効率良



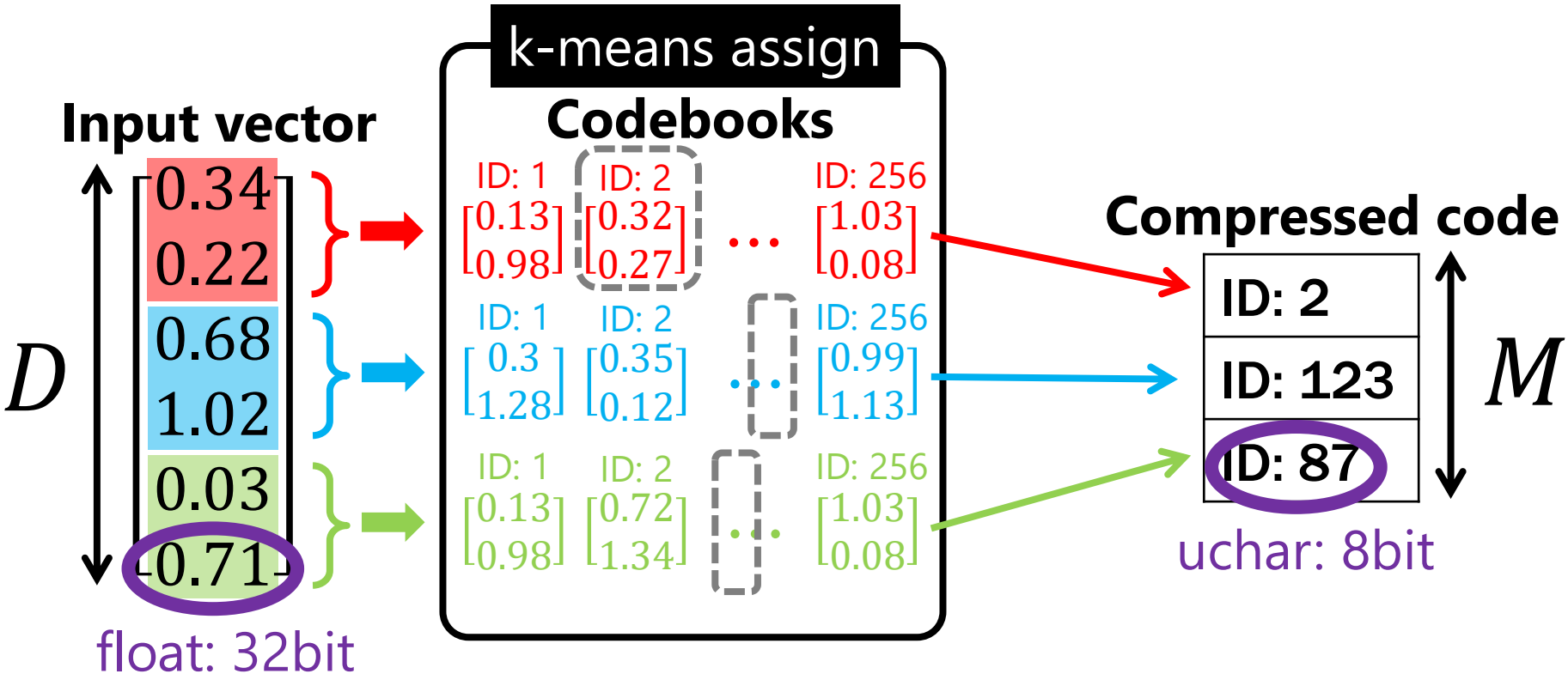
Product Quantization: メモリ効率良



e.g., $D = 128$

$128 \times 32 = 4096$ [bit]

Product Quantization: メモリ効率良



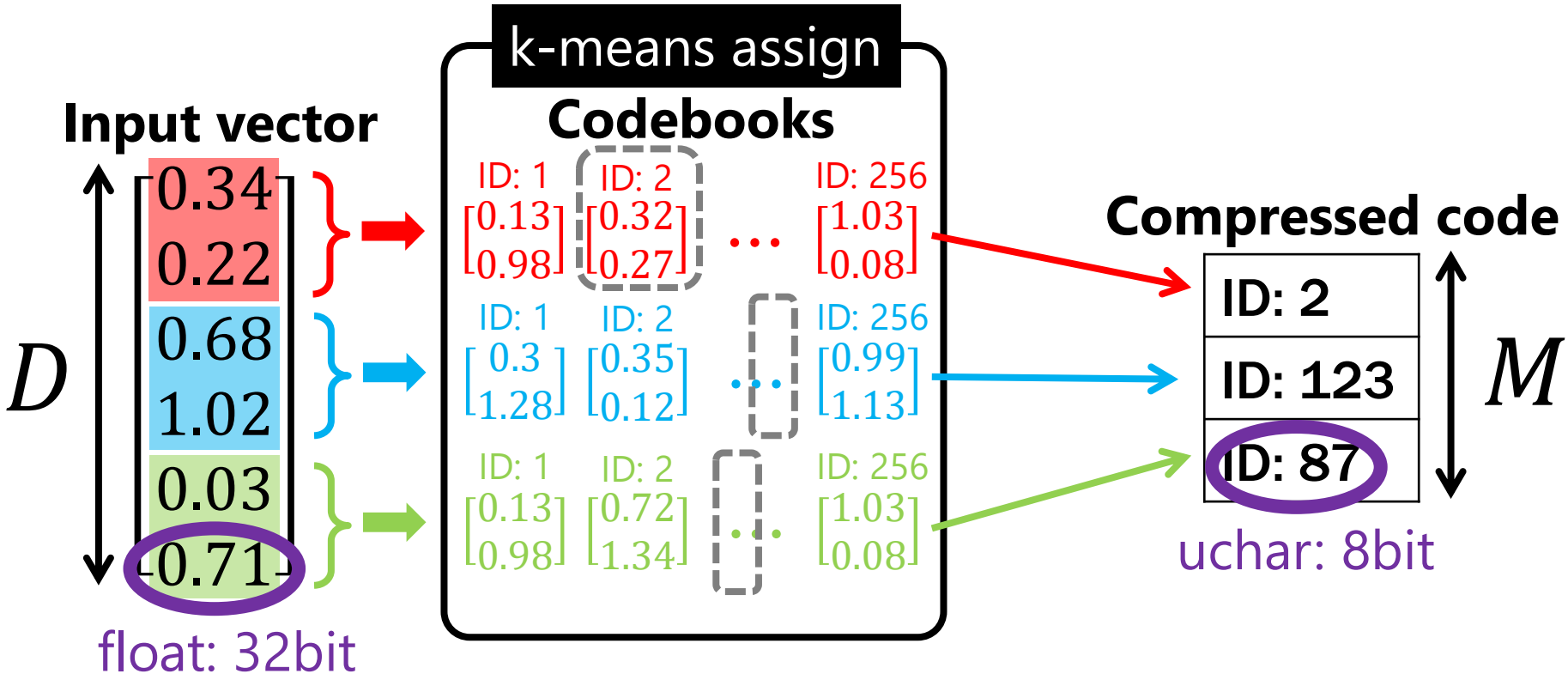
e.g., $D = 128$

$128 \times 32 = 4096$ [bit]

e.g., $M = 8$

$8 \times 8 = 64$ [bit]

Product Quantization: メモリ効率良



e.g., $D = 128$

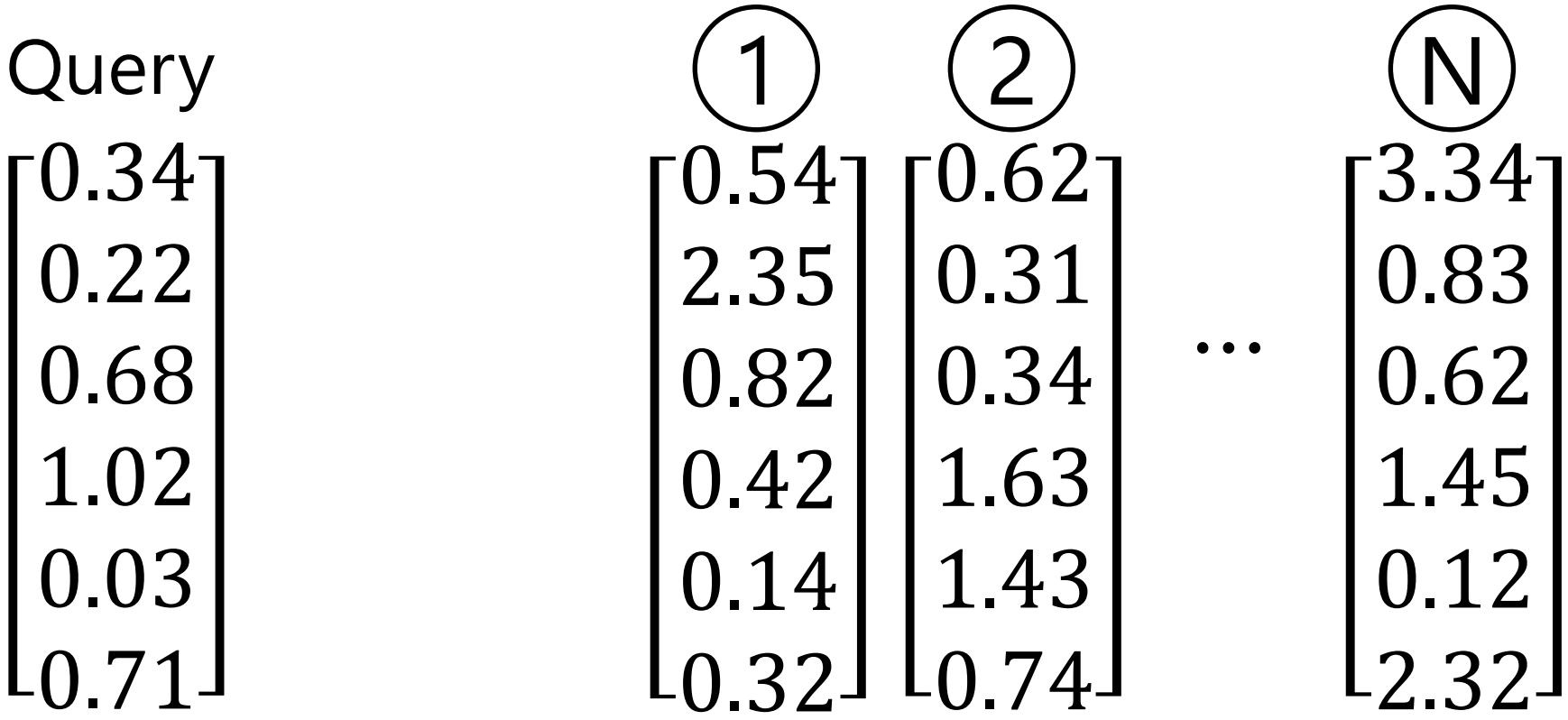
$128 \times 32 = 4096$ [bit]

e.g., $M = 8$

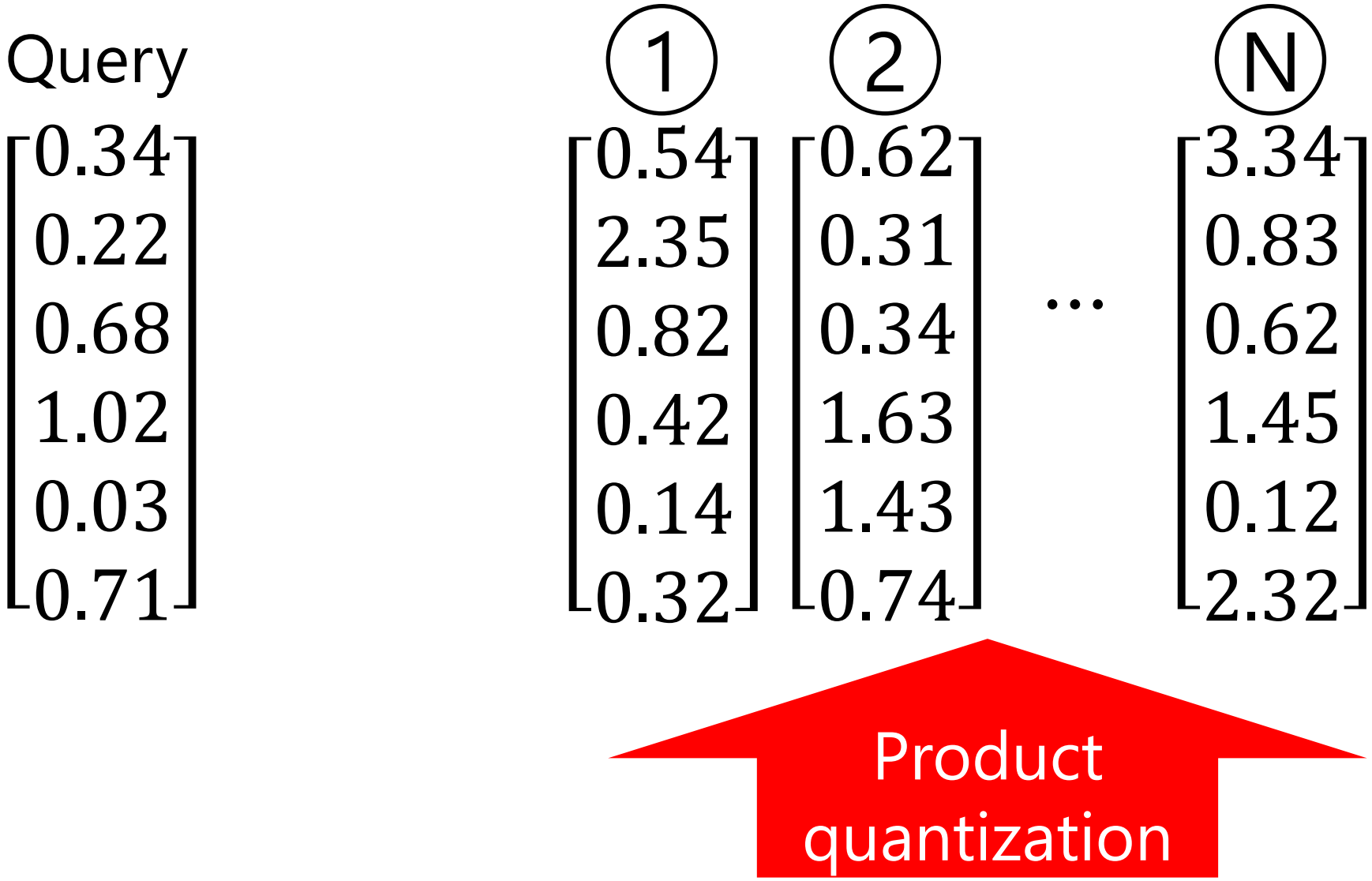
$8 \times 8 = 64$ [bit]

1/64に圧縮

Product Quantization: 距離近似



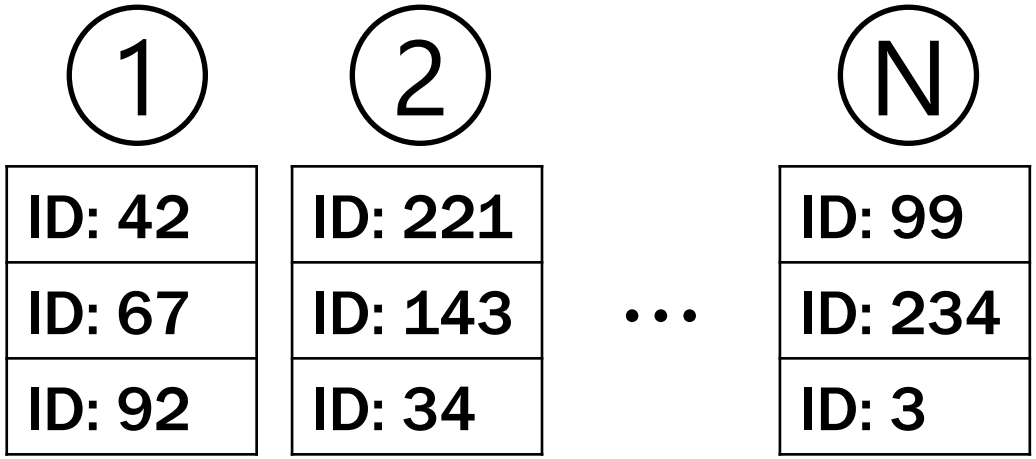
Product Quantization: 距離近似



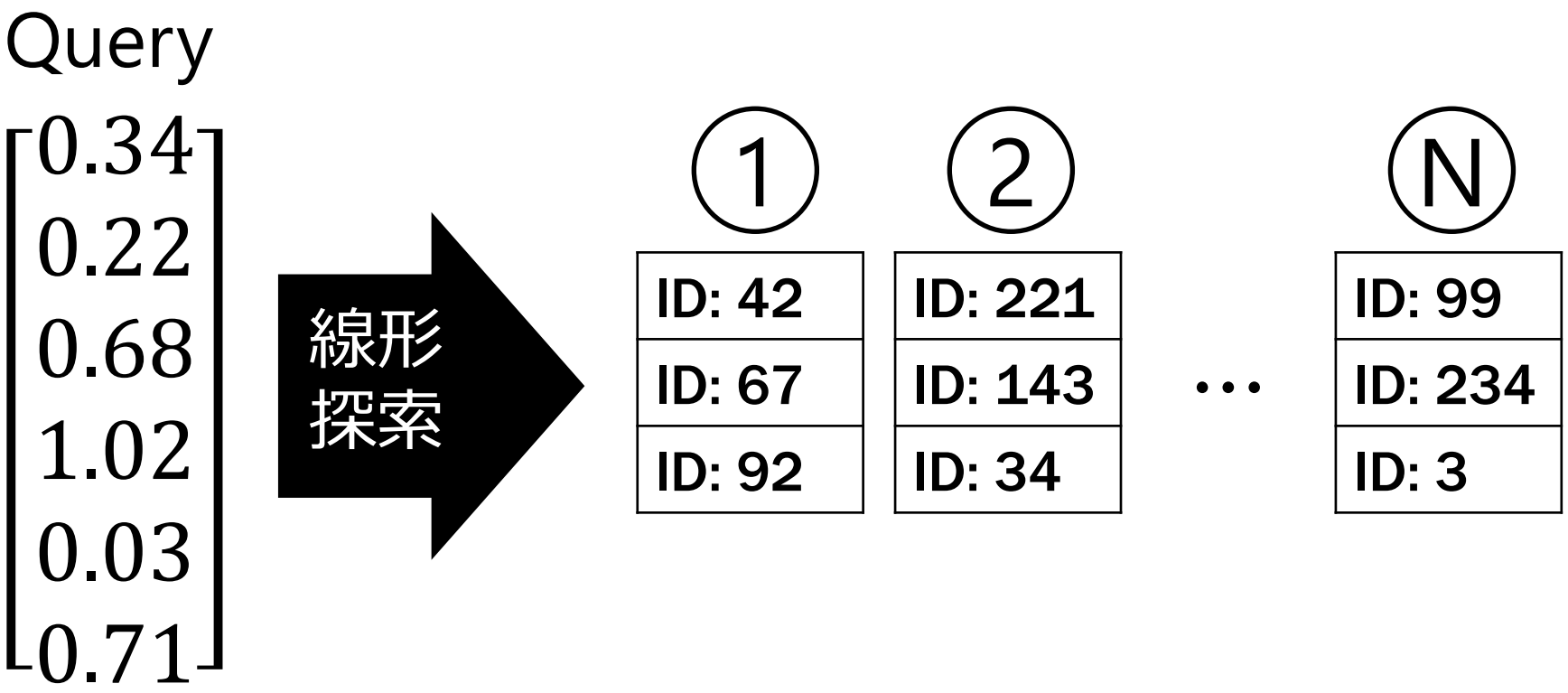
Product Quantization: 距離近似

Query

$$\begin{bmatrix} 0.34 \\ 0.22 \\ 0.68 \\ 1.02 \\ 0.03 \\ 0.71 \end{bmatrix}$$

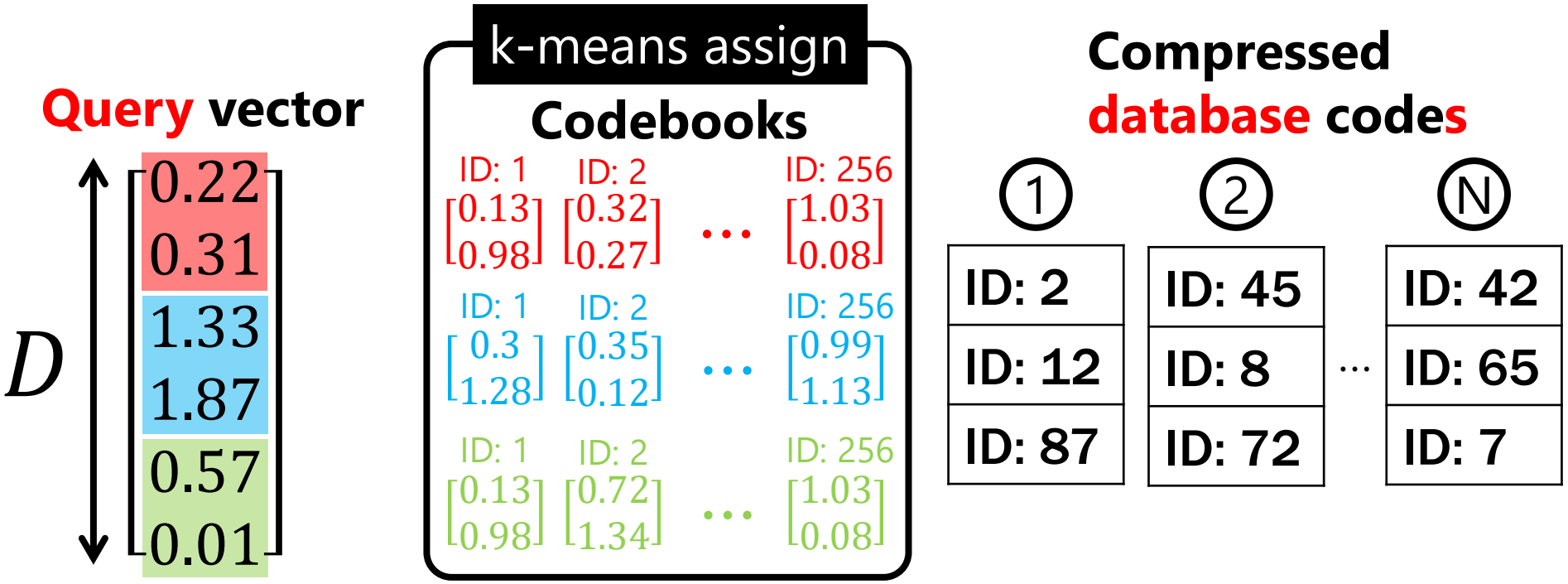


Product Quantization: 距離近似

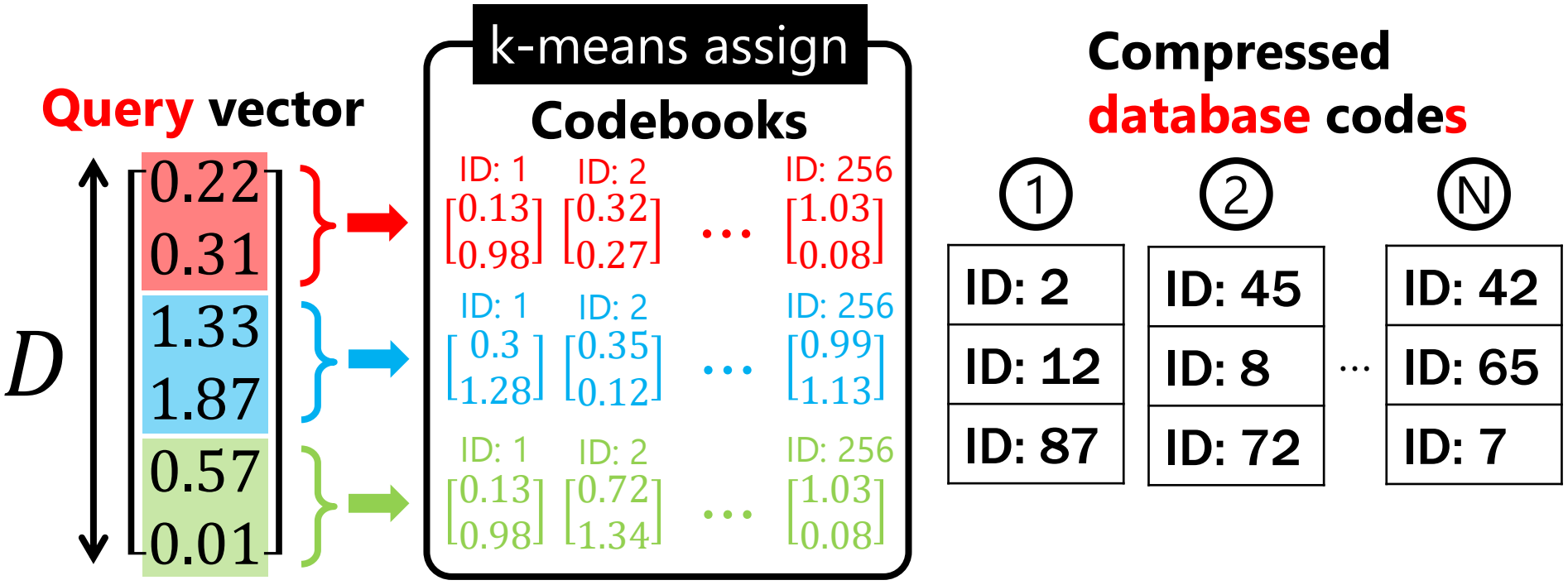


（近似的距離で）線形探索が出来る

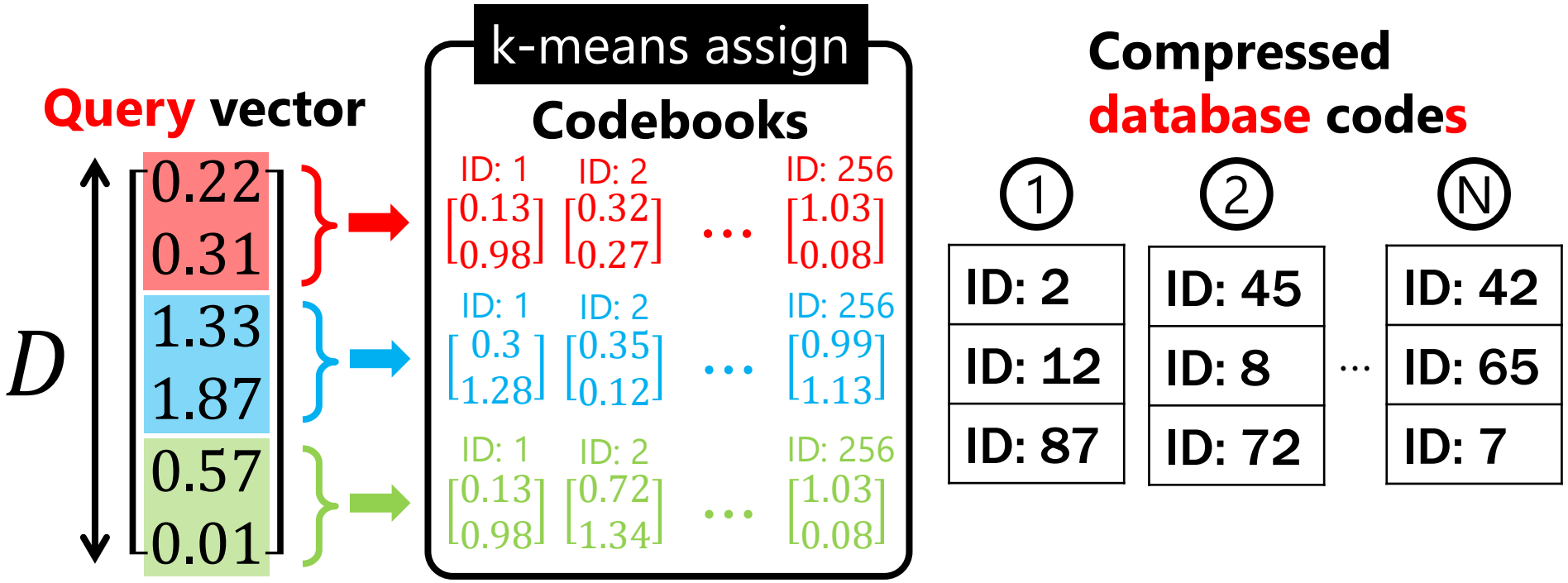
Product Quantization: 距離近似



Product Quantization: 距離近似



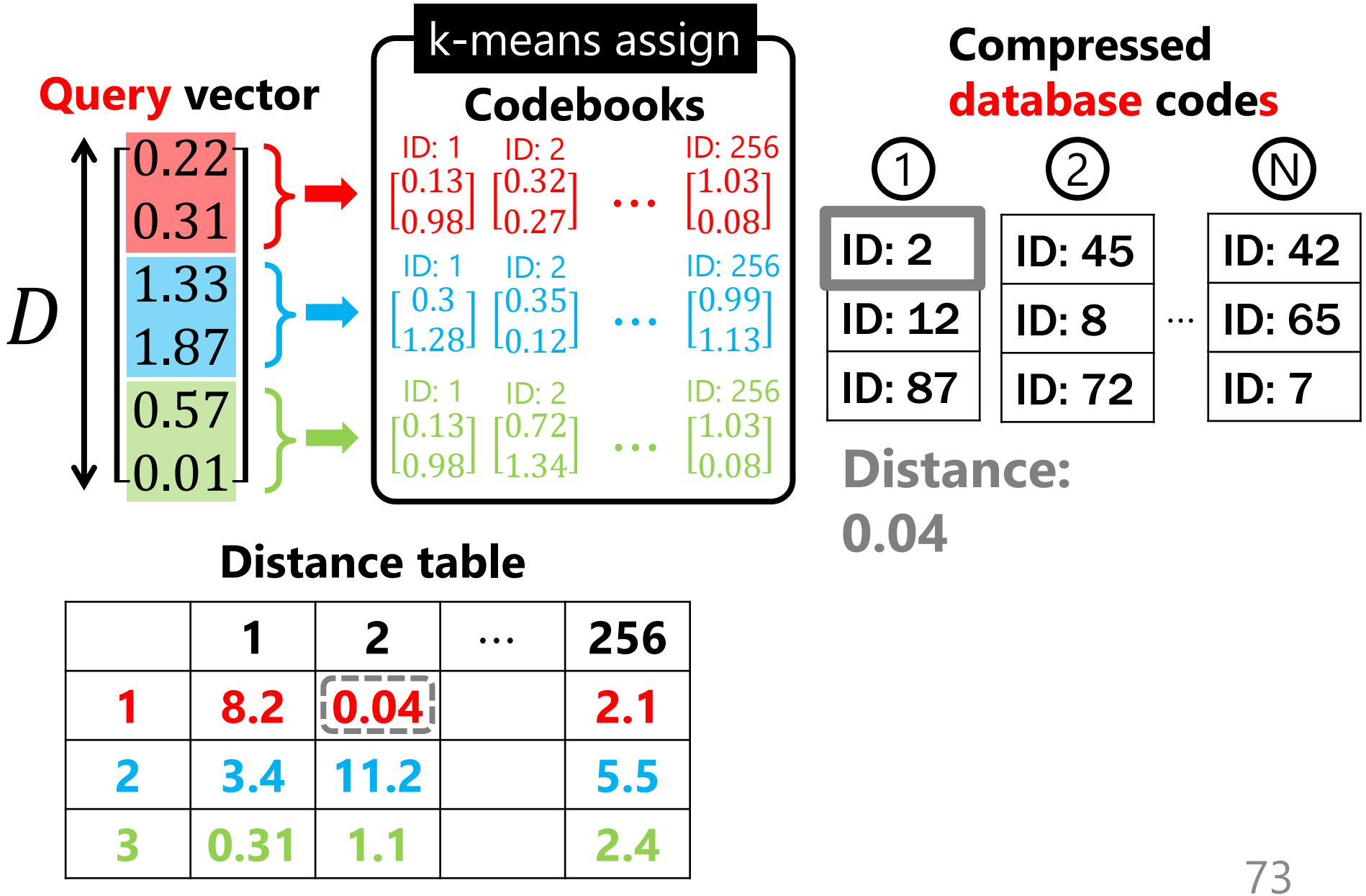
Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

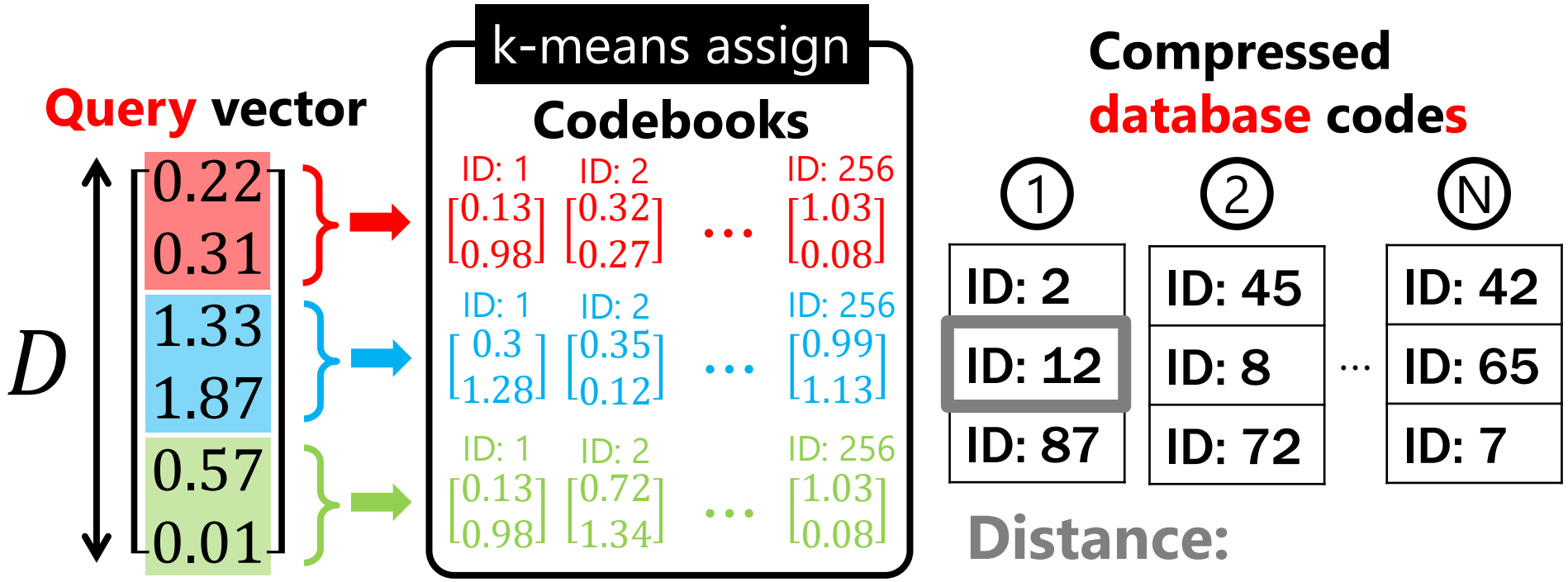
Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

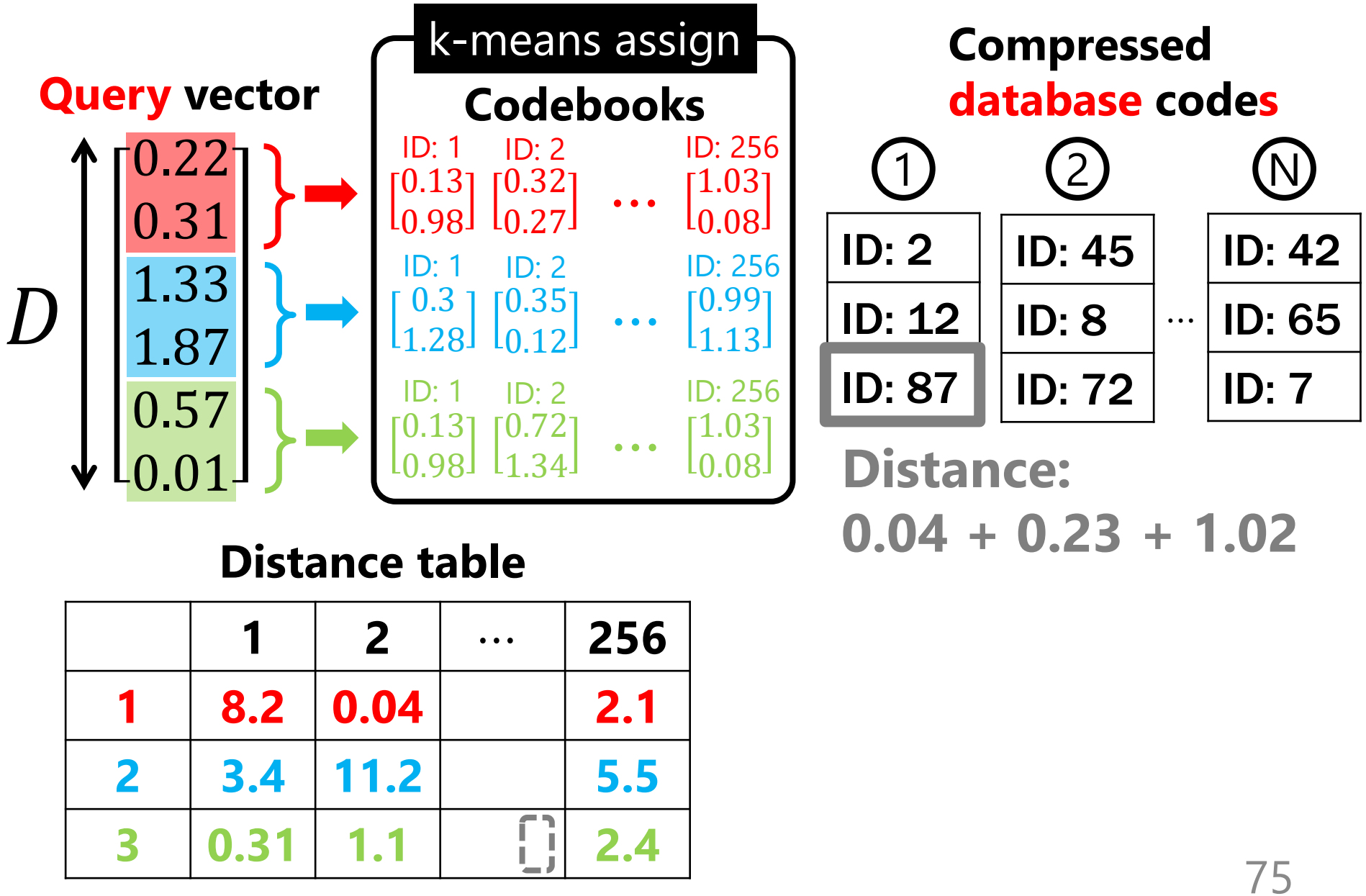
Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

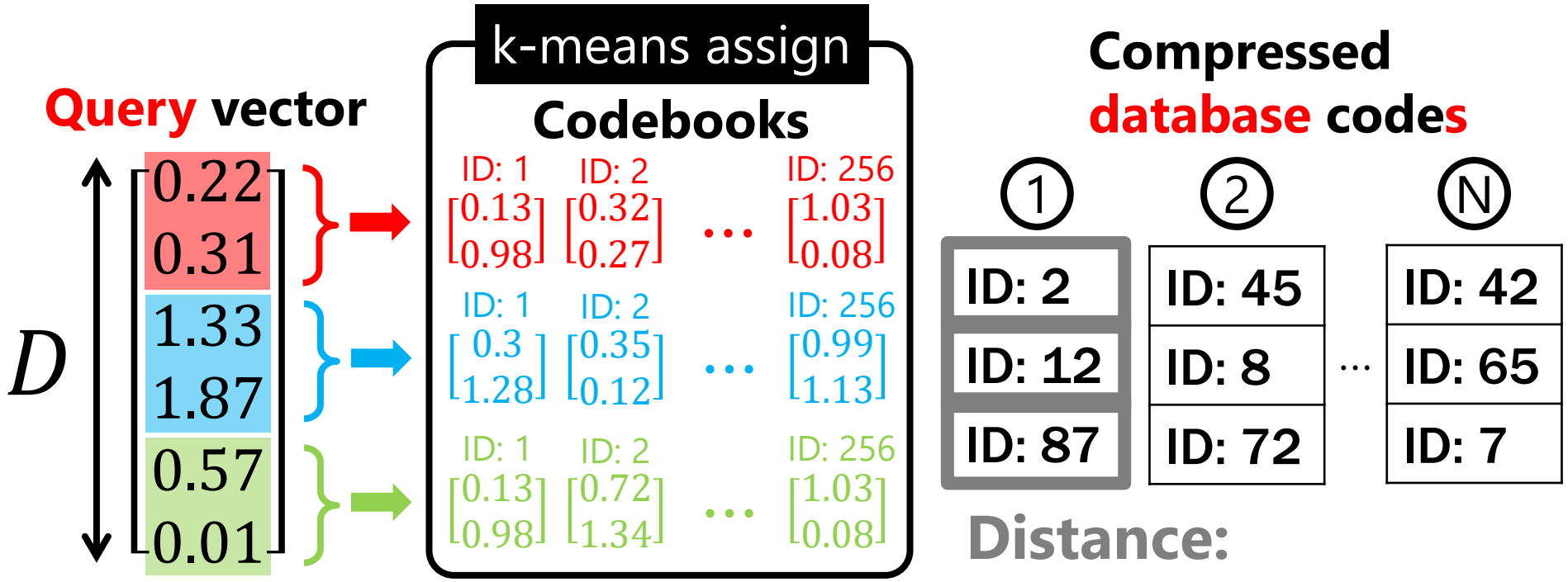
Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

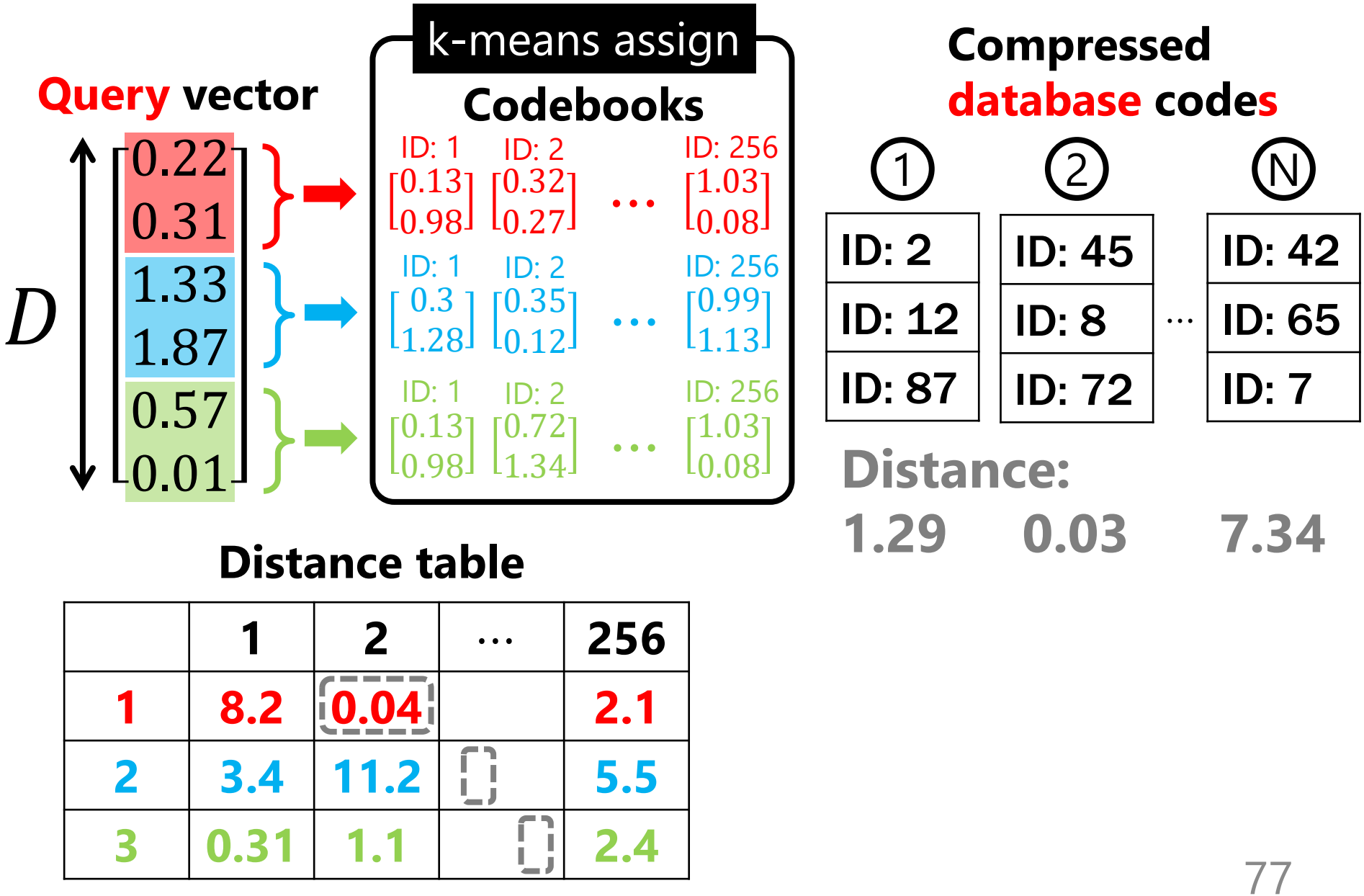
Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

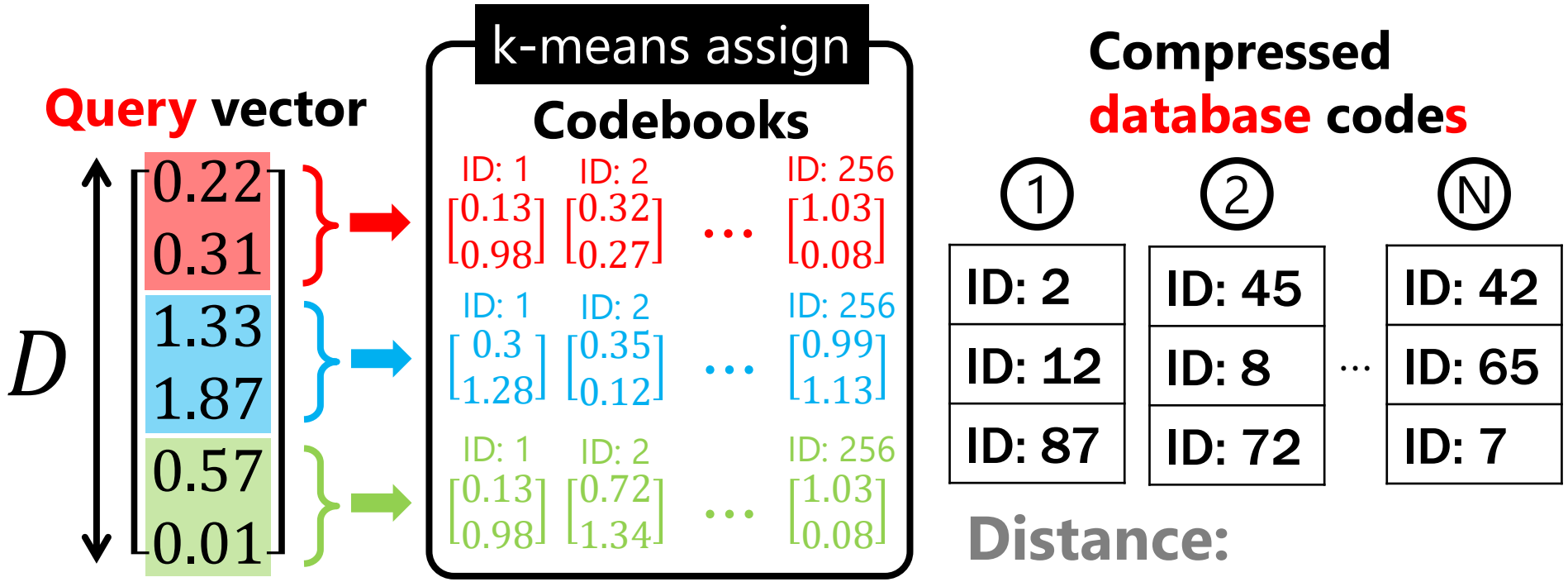
Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

Product Quantization: 距離近似



Distance table

	1	2	…	256
1	8.2	0.04		2.1
2	3.4	11.2		5.5
3	0.31	1.1		2.4

早い:

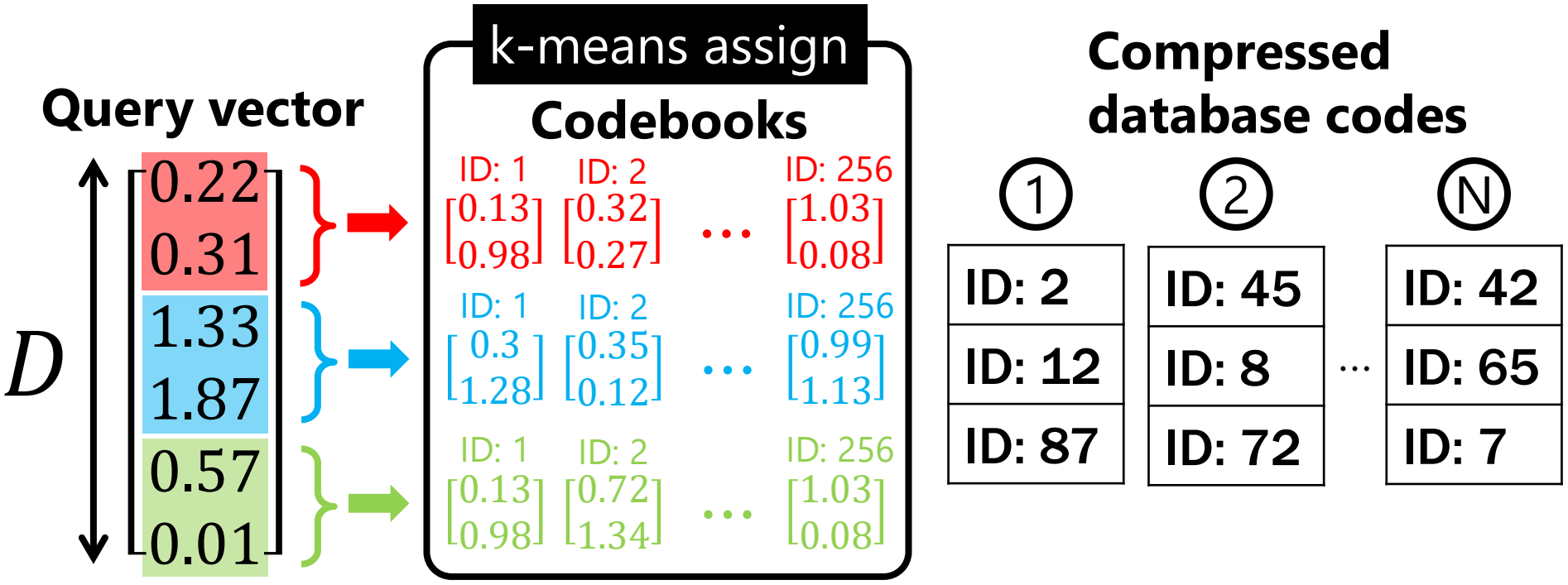
- Table lookup

計算量：
$O(DK + MN)$

正確に近似:

- 空間を 2^{8^M} に分割

Product Quantization



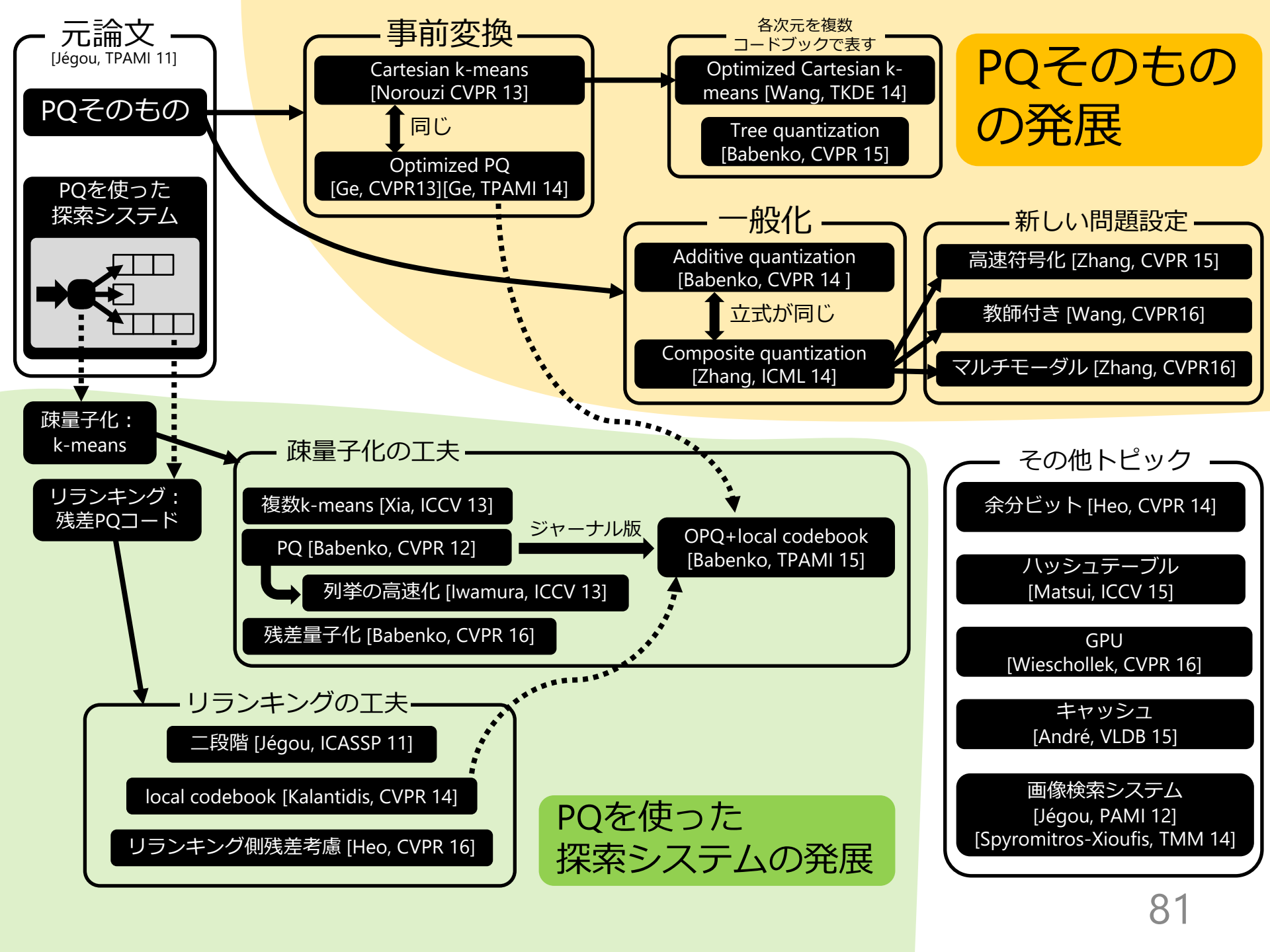
- 単純
- メモリ効率良い
- 距離 $d(input, code)^2$ を近似計算可能

Product Quantization

```python
import numpy as np
from scipy.cluster.vq import vq, kmeans2
from scipy.spatial.distance import cdist

def train(vec, M):
    Ds = int(vec.shape[1] / M) # Ds = D / M
    # codeword[m][k] = c_k^m
    codeword = np.empty((M, 256, Ds), np.float32)

    for m in range(M):
        vec_sub = vec[:, m * Ds : (m + 1) * Ds]
        codeword[m], label = kmeans2(vec_sub, 256)

    return codeword

def encode(codeword, vec): # vec = {x_n}_{n=1}^N
    M, _K, Ds = codeword.shape
    # pqcode[n] = i(x_n),  pqcode[n][m] = i^m(x_n)
    pqcode = np.empty((vec.shape[0], M), np.uint8)

    for m in range(M): # Eq. (2) and Eq. (3)
        vec_sub = vec[:, m * Ds: (m + 1) * Ds]
        pqcode[:, m], dist = vq(vec_sub, codeword[m])

    return pqcode
```

```python
def search(codeword, pqcode, query):
    M, _K, Ds = codeword.shape
    # dist_table = D(m, k)
    dist_table = np.empty((M, 256), np.float32)

    for m in range(M):
        query_sub = query[m * Ds: (m + 1) * Ds]
        dist_table[m, :] = cdist([query_sub],
      ↪  codeword[m], 'sqeuclidean')[0] # Eq. (5)

    # Eq. (6)
    dist = np.sum(dist_table[range(M), pqcode], axis=1)

    return dist

if __name__ == "__main__":
    # Read vec_train, vec ({x_n}_{n=1}^N), and query (y)
    M = 4
    codeword = train(vec_train, M)
    pqcode = encode(codeword, vec)
    dist = search(codeword, pqcode, query)
    print(dist)
```

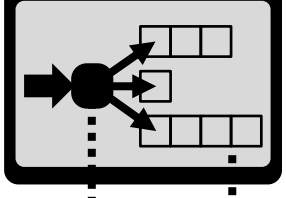
- 非常に単純


元論文
[Jégou, TPAMI 11]
PQそのもの
PQを使った
探索システム
事前変換
Cartesian k-means
[Norouzi CVPR 13]
同じ
Optimized PQ
[Ge, CVPR13][Ge, TPAMI 14]
各次元を複数
コードブックで表す
Optimized Cartesian k-means [Wang, TKDE 14]
Tree quantization
[Babenko, CVPR 15]
PQそのものの発展
一般化
Additive quantization
[Babenko, CVPR 14]
立式が同じ
Composite quantization
[Zhang, ICML 14]
新しい問題設定
高速符号化 [Zhang, CVPR 15]
教師付き [Wang, CVPR16]
マルチモーダル [Zhang, CVPR16]
疎量子化：
k-means
リランキング：
残差PQコード
疎量子化の工夫
複数k-means [Xia, ICCV 13]
PQ [Babenko, CVPR 12]
ジャーナル版
OPQ+local codebook
[Babenko, TPAMI 15]
列挙の高速化 [Iwamura, ICCV 13]
残差量子化 [Babenko, CVPR 16]
リランキングの工夫
二段階 [Jégou, ICASSP 11]
local codebook [Kalantidis, CVPR 14]
リランキング側残差考慮 [Heo, CVPR 16]
その他トピック
余分ビット [Heo, CVPR 14]
ハッシュテーブル
[Matsui, ICCV 15]
GPU
[Wieschollek, CVPR 16]
キャッシュ
[André, VLDB 15]
画像検索システム
[Jégou, PAMI 12]
[Spyromitros-Xioufis, TMM 14]
PQを使った
探索システムの発展

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った探索システム



事前変換

Cartesian k-means [Norouzi CVPR 13]

同じ

Optimized PQ [Ge, CVPR13][Ge, TPAMI 14]

各次元を複数コードブックで表す

Optimized Cartesian k-means [Wang, TKDE 14]

Tree quantization [Babenko, CVPR 15]

PQそのものの発展

一般化

Additive quantization [Babenko, CVPR 14]

立式が同じ

Composite quantization [Zhang, ICML 14]

新しい問題設定

高速符号化 [Zhang, CVPR 15]

教師付き [Wang, CVPR16]

マルチモーダル [Zhang, CVPR16]

疎量子化：k-means

リランキング：残差PQコード

疎量子化の工夫

複数k-means [Xia, ICCV 13]

PQ [Babenko, CVPR 12]

ジャーナル版

OPQ+local codebook [Babenko, TPAMI 15]

列挙の高速化 [Iwamura, ICCV 13]

残差量子化 [Babenko, CVPR 16]

リランキングの工夫

二段階 [Jégou, ICASSP 11]

local codebook [Kalantidis, CVPR 14]

リランキング側残差考慮 [Heo, CVPR 16]

その他トピック

余分ビット [Heo, CVPR 14]

ハッシュテーブル [Matsui, ICCV 15]

GPU [Wieschollek, CVPR 16]

キャッシュ [André, VLDB 15]

画像検索システム [Jégou, PAMI 12] [Spyromitros-Xioufis, TMM 14]

PQを使った探索システムの発展

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った
探索システム

事前変換

Cartesian k-means
[Norouzi CVPR 13]

同じ

Optimized PQ
[Ge, CVPR13][Ge, TPAMI 14]

各次元を複数
コードブックで表す

Optimized Cartesian k-means [Wang, TKDE 14]

Tree quantization
[Babenko, CVPR 15]

PQそのものの発展

一般化

Additive quantization
[Babenko, CVPR 14]

立式が同じ

新しい問題設定

高速符号化 [Zhang, CVPR 15]

教師付き [Wang, CVPR16]

- 「ベクトルの分割」が単純すぎないか？
- 事前にベクトルを“回転”させ，
  分割したときのエラーを最小にする

1) $\boldsymbol{x} \leftarrow R\boldsymbol{x}$

2) $\boldsymbol{x}$にPQ

このときの誤差が
最小となるように

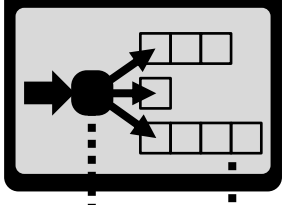
回転行列Rを求める

PQを使った
探索システムの発展

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った
探索システム



事前変換

Cartesian k-means
[Norouzi CVPR 13]

同じ

Optimized PQ
[Ge, CVPR13][Ge, TPAMI 14]

各次元を複数
コードブックで表す

Optimized Cartesian k-means [Wang, TKDE 14]

Tree quantization
[Babenko, CVPR 15]

PQそのものの発展

一般化

Additive quantization
[Babenko, CVPR 14]

立式が同じ

Composite quantization
[Zhang, ICML 14]

新しい問題設定

高速符号化 [Zhang, CVPR 15]

教師付き [Wang, CVPR16]

マルチモーダル [Zhang, CVPR16]

疎量子化：
k-means

リランキング：
残差PQコード

疎量子化の工夫

複数k-means [Xia, ICCV 13]

PQ [Babenko, CVPR 12]

ジャーナル版

OPQ+local codebook
[Babenko, TPAMI 15]

列挙の高速化 [Iwamura, ICCV 13]

残差量子化 [Babenko, CVPR 16]

リランキングの工夫

二段階 [Jégou, ICASSP 11]

local codebook [Kalantidis, CVPR 14]

リランキング側残差考慮 [Heo, CVPR 16]

その他トピック

余分ビット [Heo, CVPR 14]

ハッシュテーブル
[Matsui, ICCV 15]

GPU
[Wieschollek, CVPR 16]

キャッシュ
[André, VLDB 15]

画像検索システム
[Jégou, PAMI 12]
[Spyromitros-Xioufis, TMM 14]

PQを使った
探索システムの発展

- PQはxをM個のD/M次元コードワードで表した

$$\begin{bmatrix}0.22\\0.31\\1.33\\1.87\\0.57\\0.01\end{bmatrix} \sim \begin{bmatrix}0.32\\0.27\\\mathbf{0}\\\mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\1.19\\1.66\\\mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\\mathbf{0}\\0.32\\1.10\end{bmatrix}$$

- AQ・CQはM個のD次元コードワードで表す

$$\begin{bmatrix}0.22\\0.31\\1.33\\1.87\\0.57\\0.01\end{bmatrix} \sim \begin{bmatrix}0.02\\0.13\\0.42\\0.33\\0.12\\0.00\end{bmatrix} + \begin{bmatrix}0.19\\0.02\\0.31\\0.01\\0.23\\0.01\end{bmatrix} + \begin{bmatrix}0.02\\0.20\\0.56\\1.02\\0.66\\0.10\end{bmatrix}$$

PQの一般化．表現能力高い

PQそのものの発展

各次元を複数コードブックで表す
- Optimized Cartesian k-means [Wang, TKDE 14]
- Tree quantization [Babenko, CVPR 15]

一般化
- Additive quantization [Babenko, CVPR 14]
- 立式が同じ
- Composite quantization [Zhang, ICML 14]

新しい問題設定
- 高速符号化 [Zhang, CVPR 15]
- 教師付き [Wang, CVPR16]
- マルチモーダル [Zhang, CVPR16]

ナル版
- OPQ+local codebook [Babenko, TPAMI 15]

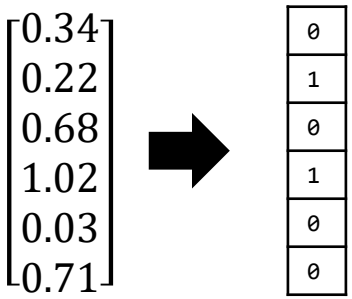
その他トピック
- 余分ビット [Heo, CVPR 14]
- ハッシュテーブル [Matsui, ICCV 15]
- GPU [Wieschollek, CVPR 16]
- キャッシュ [André, VLDB 15]
- 画像検索システム [Jégou, PAMI 12] [Spyromitros-Xioufis, TMM 14]

Qを使った
索システムの発展

ショートコードによる近似最近傍探索

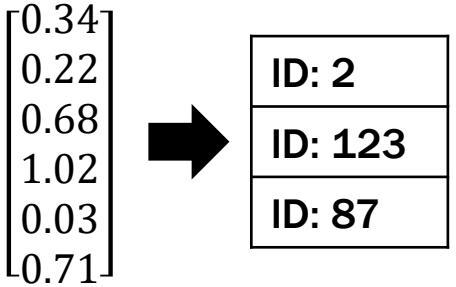
- 基本的な考え方

ハミング系



- 原理
- 手法の例
- 高速計算

ルックアップ系

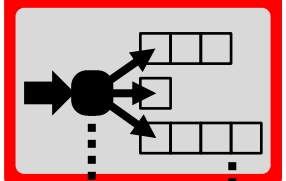


- 原理
- 探索システム

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った
探索システム



事前変換

Cartesian k-means
[Norouzi CVPR 13]

同じ

Optimized PQ
[Ge, CVPR13][Ge, TPAMI 14]

各次元を複数
コードブックで表す

Optimized Cartesian k-means [Wang, TKDE 14]

Tree quantization
[Babenko, CVPR 15]

PQそのものの発展

一般化

Additive quantization
[Babenko, CVPR 14]

立式が同じ

Composite quantization
[Zhang, ICML 14]

新しい問題設定

高速符号化 [Zhang, CVPR 15]

教師付き [Wang, CVPR16]

マルチモーダル [Zhang, CVPR16]

疎量子化：
k-means

リランキング：
残差PQコード

疎量子化の工夫

複数k-means [Xia, ICCV 13]

PQ [Babenko, CVPR 12]

ジャーナル版

OPQ+local codebook
[Babenko, TPAMI 15]

列挙の高速化 [Iwamura, ICCV 13]

残差量子化 [Babenko, CVPR 16]

リランキングの工夫

二段階 [Jégou, ICASSP 11]

local codebook [Kalantidis, CVPR 14]

リランキング側残差考慮 [Heo, CVPR 16]

PQを使った
探索システムの発展

その他トピック

余分ビット [Heo, CVPR 14]

ハッシュテーブル
[Matsui, ICCV 15]

GPU
[Wieschollek, CVPR 16]

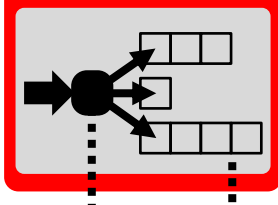
キャッシュ
[André, VLDB 15]

画像検索システム
[Jégou, PAMI 12]
[Spyromitros-Xioufis, TMM 14]

元論文
[Jégou, TPAMI 11]

PQそのもの

PQを使った
探索システム



- PQと転置インデクスを組み合わせた探索システム
- 高精度，高速，省メモリ
- この探索システムの発展形が，ユークリッド距離を用いる近似最近傍探索のstate-of-the-art
- 最も単純なIVFADCについて解説する

疎量子化：
k-means

リランキング：
残差PQコード

疎量子化の工夫

複数k-means [Xia, ICCV 13]

PQ [Babenko, CVPR 12]

ジャーナル版

OPQ+local codebook
[Babenko, TPAMI 15]

列挙の高速化 [Iwamura, ICCV 13]

残差量子化 [Babenko, CVPR 16]

リランキングの工夫

二段階 [Jégou, ICASSP 11]

local codebook [Kalantidis, CVPR 14]

リランキング側残差考慮 [Heo, CVPR 16]

PQを使った
探索システムの発展

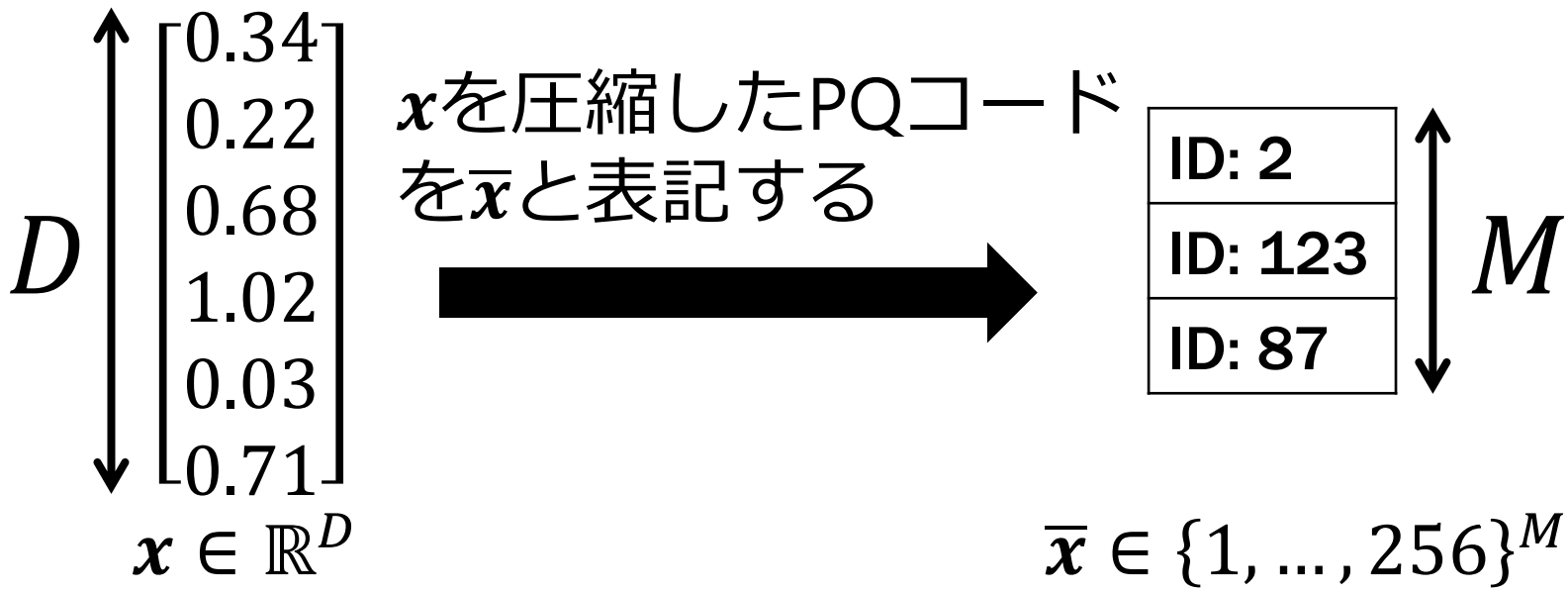
ハッシュテーブル
[Matsui, ICCV 15]

GPU
[Wieschollek, CVPR 16]

キャッシュ
[André, VLDB 15]

画像検索システム
[Jégou, PAMI 12]
[Spyromitros-Xioufis, TMM 14]

PQを用いた探索システム：復習と表記



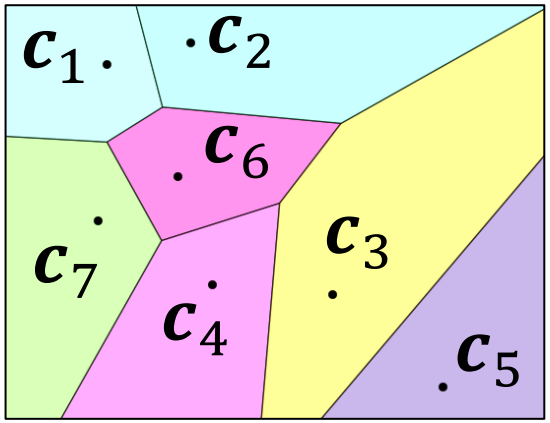
$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$として，$\boldsymbol{x}$が圧縮されて$\bar{\boldsymbol{x}}$となっているとき，
二乗距離$d(\boldsymbol{y}, \boldsymbol{x})^2$はコード$\bar{\boldsymbol{x}}$を用いて高速近似計算出来る

$$d(\boldsymbol{y}, \boldsymbol{x})^2 \sim d_A(\boldsymbol{y}, \bar{\boldsymbol{x}})^2$$

ベクトルとベクトル
の二乗距離は・・

ベクトルとコードで
近似出来る

PQを用いた探索システム：データ登録



粗量子化

$k = 1$

$k = 2$

$\vdots$

$k = K$

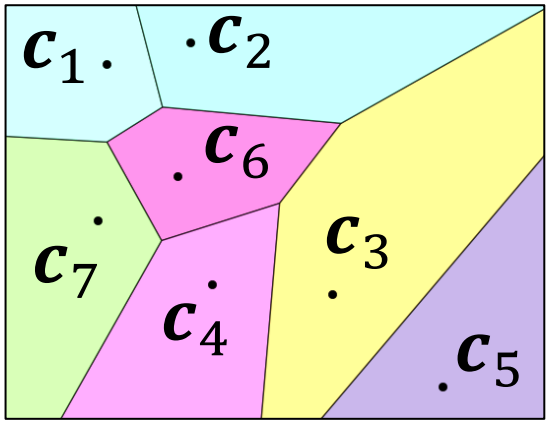
- 空間をK分割する「**粗量子化器**」を用意しておく．ここでは単純なボロノイ空間分割（k-means割り当てそのもの）
- 各$\{\boldsymbol{c}_k\}_{k=1}^{K}$は訓練データに単純にk-meansを適用し作る

PQを用いた探索システム：データ登録

- ベクトル$\boldsymbol{x}_1$の登録を考える

$$\begin{bmatrix} 1.02 \\ 0.73 \\ 0.56 \\ 1.37 \\ 1.37 \\ 0.72 \end{bmatrix}$$

$\boldsymbol{x}_1$



粗量子化

$k = 1$

$k = 2$

$\vdots$

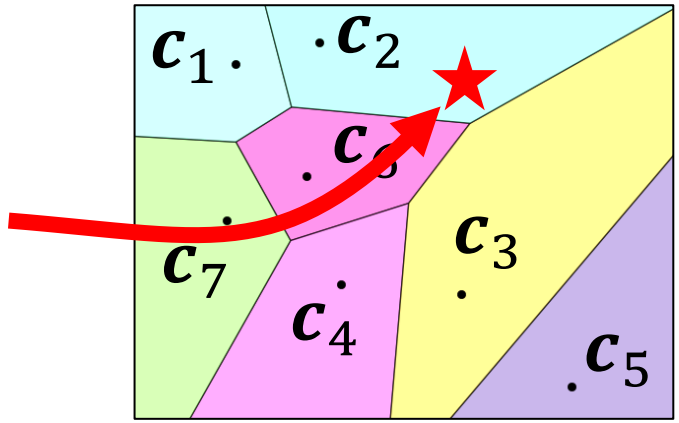
$k = K$

PQを用いた探索システム：データ登録

- ベクトル$\boldsymbol{x}_1$の登録を考える

$$\begin{bmatrix} 1.02 \\ 0.73 \\ 0.56 \\ 1.37 \\ 1.37 \\ 0.72 \end{bmatrix}$$

$\boldsymbol{x}_1$



粗量子化

$k = 1$

$k = 2$

$\vdots$

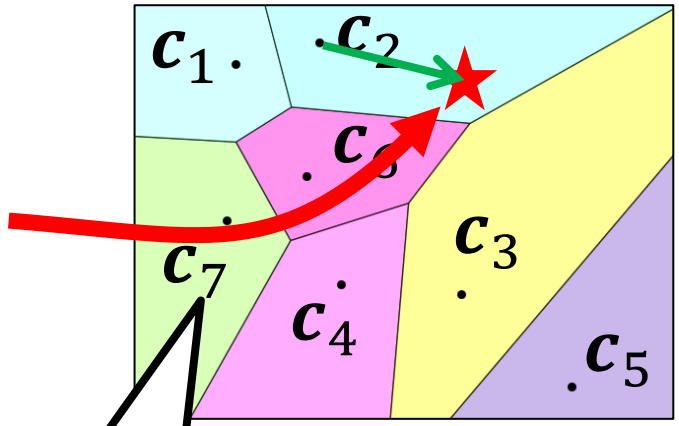
$k = K$

PQを用いた探索システム：データ登録

- ベクトル$\boldsymbol{x}_1$の登録を考える

$$\begin{bmatrix}1.02\\0.73\\0.56\\1.37\\1.37\\0.72\end{bmatrix}$$

$\boldsymbol{x}_1$



粗量子化

$k = 1$

$k = 2$

$\vdots$

$k = K$

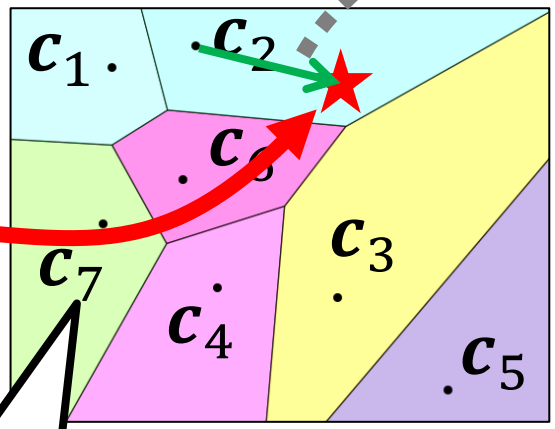
- $\boldsymbol{x}_1$に一番近いものは$\boldsymbol{c}_2$
- $\boldsymbol{x}_1$と$\boldsymbol{c}_2$の**残差** $\boldsymbol{r}_1 = \boldsymbol{x}_1 - \boldsymbol{c}_2$ (→) を計算する

PQを用いた探索システム：データ登録

- ベクトル$\boldsymbol{x}_1$の登録を考える

$$\begin{bmatrix}1.02\\0.73\\0.56\\1.37\\1.37\\0.72\end{bmatrix}$$

$\boldsymbol{x}_1$

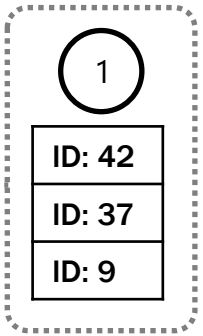


粗量子化

$k = 1$

$k = 2$

$k = K$



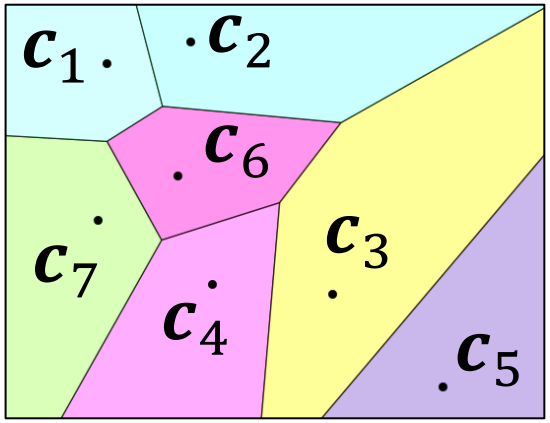
$\bar{\boldsymbol{r}}_1$

- $\boldsymbol{x}_1$に一番近いものは$\boldsymbol{c}_2$
- $\boldsymbol{x}_1$と$\boldsymbol{c}_2$の**残差** $\boldsymbol{r}_1 = \boldsymbol{x}_1 - \boldsymbol{c}_2$ (→) を計算する

- 残差 $\boldsymbol{r}_1$ をPQで圧縮し，コード$\bar{\boldsymbol{r}}_1$を作り，番号「1」とともに記録する
- すなわち，$(i, \bar{\boldsymbol{r}}_i)$を記録する

PQを用いた探索システム：データ登録

- 全データに関して，「番号＋残差」をリストとして保存



粗量子化

$k = 1$

245	12	1932	1721
ID: 42	ID: 25	ID: 38	ID: 16
ID: 37	ID: 47	ID: 49	ID: 72
ID: 9	ID: 32	ID: 72	ID: 95

$k = 2$

1	3721
ID: 42	ID: 18
ID: 37	ID: 4
ID: 9	ID: 96

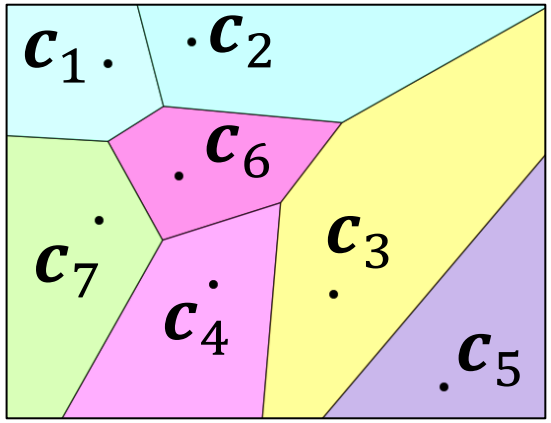
$k = K$

8621	145	324
ID: 24	ID: 77	ID: 32
ID: 54	ID: 21	ID: 11
ID: 23	ID: 5	ID: 85

PQを用いた探索システム：探索

- クエリ$\boldsymbol{q}$に近いデータを探す

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



粗量子化

$k = 1$

245	12	1932	1721
ID: 42	ID: 25	ID: 38	ID: 16
ID: 37	ID: 47	ID: 49	ID: 72
ID: 9	ID: 32	ID: 72	ID: 95

$k = 2$

1	3721
ID: 42	ID: 18
ID: 37	ID: 4
ID: 9	ID: 96

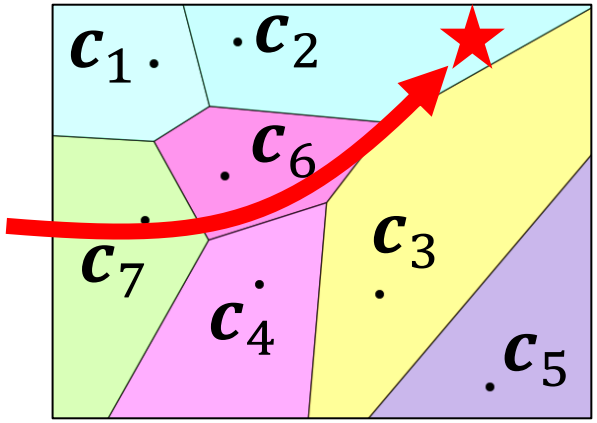
$k = K$

8621	145	324
ID: 24	ID: 77	ID: 32
ID: 54	ID: 21	ID: 11
ID: 23	ID: 5	ID: 85

PQを用いた探索システム：探索

- クエリqに近いデータを探す

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



粗量子化

$k = 1$

245	12	1932	1721
ID: 42	ID: 25	ID: 38	ID: 16
ID: 37	ID: 47	ID: 49	ID: 72
ID: 9	ID: 32	ID: 72	ID: 95

$k = 2$

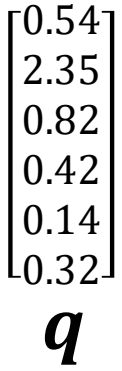
1	3721
ID: 42	ID: 18
ID: 37	ID: 4
ID: 9	ID: 96

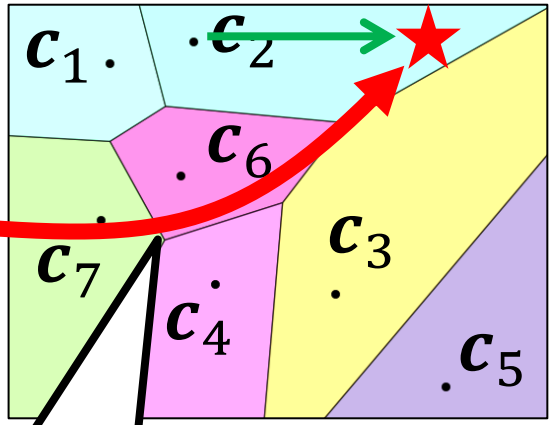
$k = K$

8621	145	324
ID: 24	ID: 77	ID: 32
ID: 54	ID: 21	ID: 11
ID: 23	ID: 5	ID: 85

PQを用いた探索システム：探索

- クエリqに近いデータを探す





粗量子化

- qに一番近いものはc_2
- qとc_2の**残差** $r_q = q - c_2$ を計算する

$k = 1$

245	12	1932	1721
ID: 42	ID: 25	ID: 38	ID: 16
ID: 37	ID: 47	ID: 49	ID: 72
ID: 9	ID: 32	ID: 72	ID: 95

$k = 2$

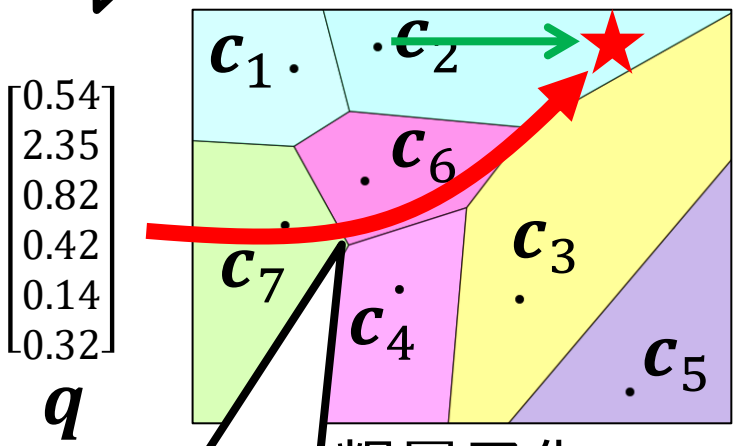
1	3721
ID: 42	ID: 18
ID: 37	ID: 4
ID: 9	ID: 96

$k = K$

8621	145	324
ID: 24	ID: 77	ID: 32
ID: 54	ID: 21	ID: 11
ID: 23	ID: 5	ID: 85

PQを用いた探索システム：探索

- クエリ$\boldsymbol{q}$に近いデータを探す



- $\boldsymbol{q}$に一番近いものは$\boldsymbol{c}_2$
- $\boldsymbol{q}$と$\boldsymbol{c}_2$の**残差** $\boldsymbol{r}_q = \boldsymbol{q} - \boldsymbol{c}_2$ を計算する

$k = 1$

245	12	1932	1721
ID: 42	ID: 25	ID: 38	ID: 16
ID: 37	ID: 47	ID: 49	ID: 72
ID: 9	ID: 32	ID: 72	ID: 95

$k = 2$

1	3721
ID: 42	ID: 18
ID: 37	ID: 4
ID: 9	ID: 96

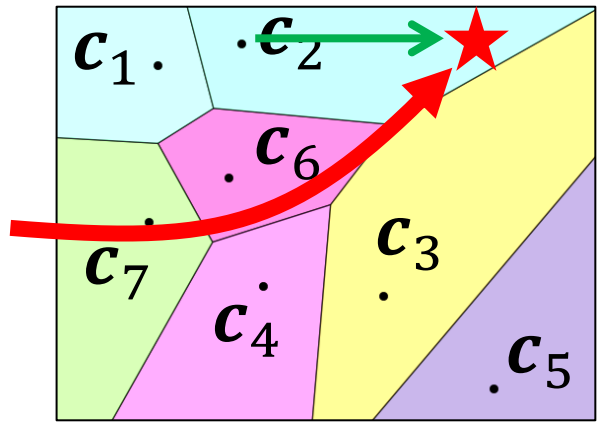
- $k = 2$に登録されている各$(i, \bar{\boldsymbol{r}}_i)$について，$\boldsymbol{r}_q$と比べる

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

- 最も近いものを選ぶ（リランキング．戦略色々）

PQを用いた探索システム：探索

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



粗量子化

- 粗量子化のコスト＋リランキングのコストを調整することで，高速な探索を実現
- 残差に注目することで，近似精度を高めた

$k = 1$

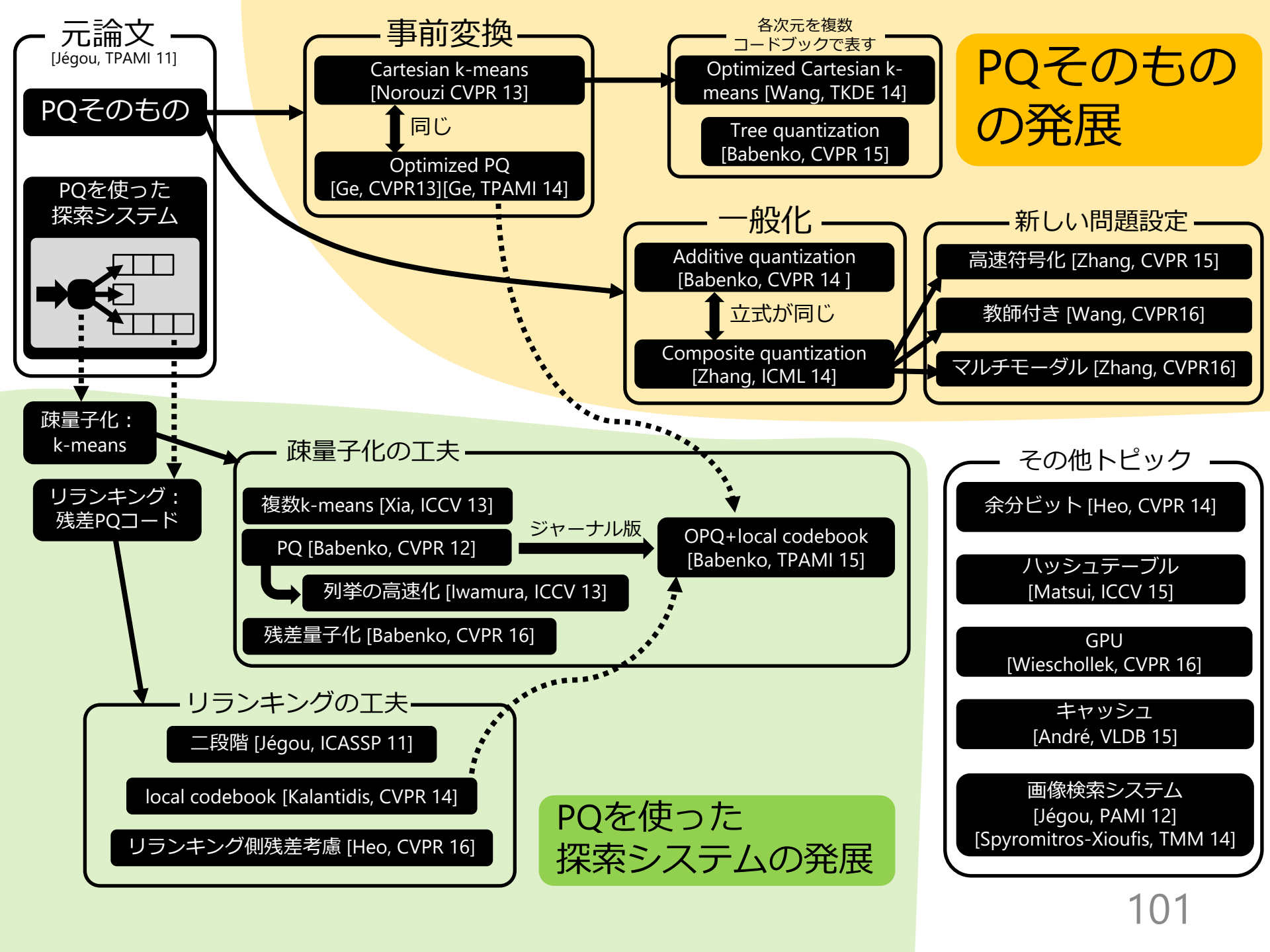
245	12	1932	1721
ID: 42	ID: 25	ID: 38	ID: 16
ID: 37	ID: 47	ID: 49	ID: 72
ID: 9	ID: 32	ID: 72	ID: 95

$k = 2$

1	3721
ID: 42	ID: 18
ID: 37	ID: 4
ID: 9	ID: 96

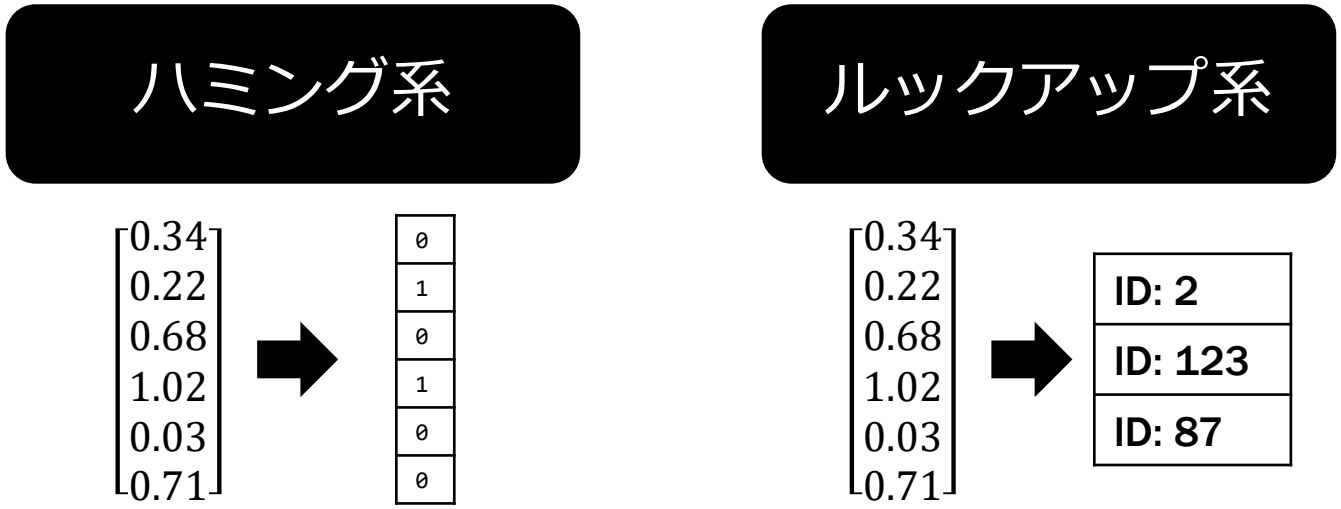
$k = K$

8621	145	324
ID: 24	ID: 77	ID: 32
ID: 54	ID: 21	ID: 11
ID: 23	ID: 5	ID: 85


元論文
[Jégou, TPAMI 11]
PQそのもの
PQを使った
探索システム
事前変換
Cartesian k-means
[Norouzi CVPR 13]
同じ
Optimized PQ
[Ge, CVPR13][Ge, TPAMI 14]
各次元を複数
コードブックで表す
Optimized Cartesian k-means [Wang, TKDE 14]
Tree quantization
[Babenko, CVPR 15]
PQそのものの発展
一般化
Additive quantization
[Babenko, CVPR 14]
立式が同じ
Composite quantization
[Zhang, ICML 14]
新しい問題設定
高速符号化 [Zhang, CVPR 15]
教師付き [Wang, CVPR16]
マルチモーダル [Zhang, CVPR16]
疎量子化：
k-means
リランキング：
残差PQコード
疎量子化の工夫
複数k-means [Xia, ICCV 13]
PQ [Babenko, CVPR 12]
ジャーナル版
OPQ+local codebook
[Babenko, TPAMI 15]
列挙の高速化 [Iwamura, ICCV 13]
残差量子化 [Babenko, CVPR 16]
リランキングの工夫
二段階 [Jégou, ICASSP 11]
local codebook [Kalantidis, CVPR 14]
リランキング側残差考慮 [Heo, CVPR 16]
その他トピック
余分ビット [Heo, CVPR 14]
ハッシュテーブル
[Matsui, ICCV 15]
GPU
[Wieschollek, CVPR 16]
キャッシュ
[André, VLDB 15]
画像検索システム
[Jégou, PAMI 12]
[Spyromitros-Xioufis, TMM 14]
PQを使った
探索システムの発展

ショートコードによる近似最近傍探索

- 基本的な考え方：ベクトルを省メモリ表現に圧縮し圧縮された世界で探索を行う



ベクトル表現	バイナリコード：$\{0,1\}^B$	PQコード：$\{1,\dots,256\}^M$
距離表現	ハミング距離	表参照距離
表現能力	○	◎
距離計算速度	◎	○
Pros	補助構造がいらない	元のベクトルを 近似再構成可能
Cons	近似再構成は出来ない	補助構造が必要 （コードブック）

参考文献

[Weiss, NIPS 08] Y. Weiss, A. Torralba, and R. Fergus: “Spectral Hashing”, NIPS (2008).
[Norouzi, TPAMI 13] M. Norouzi, A. Punjani, and D. FJ. Fleet: “Fast Exact Search in Hamming Space With Multi-Index Hashing”, IEEE TPAMI (2013).
[Muja, TPAMI 14] M. Muja and D. G. Lowe: “Scalable nearest neighbour algorithms for high dimensional data”, IEEE TPAMI (2014).
[Jégou, TPAMI 11] H. Jégou, M. Douze and C. Schmid: “Product quantization for nearest neighbor search”, IEEE TPAMI (2011).
[Norouzi, CVPR 13] M. Norouzi and D. J. Fleet: “Cartesian k-means”, CVPR (2013).
[Ge, CVPR 13] T. Ge, K. He, Q. Ke and J. Sun: “Optimized product quantization for approximate nearest neighbor search”, CVPR (2013).
[Ge, TPAMI 14] T. Ge, K. He, Q. Ke and J. Sun: “Optimized product quantization”, IEEE TPAMI, (2014).
[Wang, TKDE 14] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen and S. Li: “Optimized cartesian k-means”, IEEE TKDE (2014).
[Babenko, CVPR 15] A. Babenko and V. Lempitsky: “Tree quantization for large-scale similarity search and classification”, CVPR (2015).
[Babenko, CVPR 14] A. Babenko and V. Lempitsky: “Additive quantization for extreme vector compression”, CVPR (2014).
[Zhang, ICML 14] T. Zhang, C. Du and J. Wang: “Composite quantization for approximate nearest neighbor search”, ICML (2014).
[Zhang, CVPR 15] T. Zhang, G.-J. Qi, J. Tang and J. Wang: “Sparse composite quantization”, CVPR (2015).
[Wang, CVPR16] X. Wang, T. Zhang, G.-J. Qi, J. Tang and J. Wang: “Supervised quantization for similarity search”, CVPR (2016).
[Zhang, CVPR 16] T. Zhang and J. Wang: “Collaborative quantization for cross-modal similarity search”, CVPR (2016).
[Xia, ICCV 13] Y. Xia, K. He, F. Wen and J. Sun: “Joint inverted indexing”, ICCV (2013).
[Babenko, CVPR 12] A. Babenko and V. Lempitsky: “The inverted multi-index”, CVPR (2012).
[Iwamura, ICCV 13] M. Iwamura, T. Sato and K. Kise: “What is the most efficient way to select nearest neighbor candidates for fast approximate nearest neighbor search?”, ICCV (2013).
[Babenko, CVPR 16] A. Babenko and V. Lempitsky: “Efficient indexing of billion-scale datasets of deep descriptors”, CVPR (2016).
[Babenko, TPAMI 15] A. Babenko and V. Lempitsky: “The inverted multi-index”, IEEE TPAMI, (2015).
[Jégou, ICASSP 11] H. Jégou, R. Tavenard, M. Douze and L. Amsaleg: “Searching in one billion vectors: Re-rank with souce coding”, ICASSP (2011).
[Kalantidis, CVPR 14] Y. Kalantidis and Y. Avrithis: “Locally optimized product quantization for approximate nearest neighbor search”, CVPR (2014).
[Heo, CVPR 16] J.-P. Heo, Z. Lin, X. Shen, J. Brandt and S. eui Yoon: “Shortlist selection with residual aware distance estimator for k-nearest neighbor search”, CVPR (2016).
[Heo, CVPR 14] J.-P. Heo, Z. Lin and S.-E. Yoon: “Distance encoded product quantization”, CVPR (2014).
[Matsui, ICCV 15] Y. Matsui, T. Yamasaki and K. Aizawa: “Pqtable: Fast exact asymmetric distance neighbor search for product quantization using hash tables”, ICCV (2015).
[Wieschollek, CVPR 16] P. Wieschollek, O. Wang, A. Sorkine-Hornung and H. P. A. Lensch: “Efficient large-scale approximate nearest neighbor search on the gpu”, CVPR (2016).
[André, VLDB 15] F. André, A.-M. Kermarrec and N. L. Scouarnec: “Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan”, VLDB (2015).
[Jégou, PAMI 12] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez and C. Schmid: “Aggregating local image descriptors into compact codes”, IEEE TPAMI, (2012).
[Spyromitros-Xioufis, TMM 14] E. Spyromitros-Xioufis, S. Papadopoulos, I. Y. Kompatsiaris, G. Tsoumakas and I. Vlahavas: “A comprehensive study over vlad and product quantization in large-scale image retrieval”, IEEE TMM, (2014).