2N-02

# 近似最近傍探索のためのマージ可能な転置インデックス

松岡 暉心 †　　松井 勇佑 †

東京大学 †

## 1　はじめに

最近傍探索は、クエリベクトルが与えられたとき、それに最も近いベクトルをデータベースベクトルの集合の中から探し出すための技術である。その応用範囲は広く、情報検索や画像分類、推薦システムなど、様々な分野で最近傍探索の手法が活用されている。また近年では、大規模データに対してオンメモリで探索を行うための、近似最近傍探索の手法の研究が急速に発展している。

本稿では、実用上重要であるがこれまでの研究では考慮されていなかった、複数の近傍探索インデックスをマージする操作を考える。そして、このマージ操作の後でも高速に近似最近傍探索を行うことのできるインデックスを提案する。

## 2　関連研究

本稿では近似最近傍探索のために、 [1] で提案され現在最も広く用いられている、転置インデックス (IVF) と直積量子化 (PQ) を組み合わせた IVFPQ インデックスを用いる。そこで、本章では転置インデックスと直積量子化について説明する。

### 2.1　転置インデックス (Inverted File Index, IVF)

転置インデックスは、空間を分割することにより効率的に近傍探索を行うためのデータ構造である。転置インデックスの作成では、まずトレーニングベクトルを $R$ 個のクラスタにクラスタリングすることで粗量子化コードブックを得る。そして、データベースベクトルをその粗量子化コードブックで量子化し、同じコードに量子化されたベクトルは転置インデックスの同じエントリに追加する。

探索の際は、まずクエリ $\mathbf{q} \in \mathbb{R}^D$ と $R$ 個の粗量子化コードワードとの間で距離計算を行う。次に、そのうち $\mathbf{q}$ に近い $p$ 個のコードワードに対応する転置インデックスのエントリのデータベースベクトルと、$\mathbf{q}$ との間でのみ距離計算を行う。なお、$p$ はハイパーパラメータであり、値を大きくすると精度が向上するが探索速度は低下する。

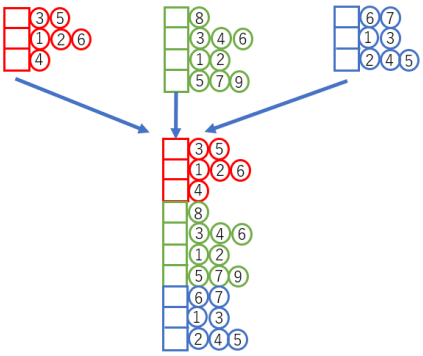

図 1　複数の転置インデックスのナイーブなマージ手法 (図中の数字はデータベースベクトルの ID)

### 2.2　直積量子化 (Product Quantization, PQ)

直積量子化は、近傍探索のためのデータ圧縮手法である。この手法では、$D$ 次元空間を $S$ 個の $D/S$ 次元部分空間の直積であると考え、それぞれの部分空間でベクトル量子化を行う。よって、トレーニングベクトルの集合は $S$ 個の部分空間について独立に $K$ 個のクラスタにクラスタリングされ、$S$ 個の直積量子化コードブックが得られる。データベースベクトルは、この $S$ 個のコードブックにより各部分空間について独立に量子化される。

ここで、クエリと直積量子化されたデータベースベクトルとの距離計算について考える。この距離計算に先立ち、クエリ $\mathbf{q} \in \mathbb{R}^D$ の各サブベクトル $\mathbf{q}^s \in \mathbb{R}^{D/S}$ ($s \in \{1, 2, ..., S\}$) について、$s$ 番目の部分空間に対応する直積量子化コードブックに含まれる $K$ 個の直積量子化コードワードとの距離をあらかじめ計算し、これを距離表として記録しておく。すると、直積量子化されたデータベースベクトル $\tilde{\mathbf{x}} = [x_1, x_2, ..., x_S]$ (各要素は 0 以上 $K$ 未満の整数) とクエリとの近似距離は、距離表を $S$ 回参照し、その値を足すことで高速に求められる。

† Kishin Matsuoka, Yusuke Matsui
† The University of Tokyo

## 3　課題

いま、複数の IVFPQ インデックスをマージし、そのマージされたインデックスに対して探索を行うことを考える。この際、インデックスをマージするためのナイーブな手法は、図 1 のように転置インデックスをそのまま結合する方法である。しかし、この手法では、1 つのクエリに対してインデックスの個数分だけ距離表を計算しなければならない。そのため、インデックスの個数の増加に伴って、距離表計算にかかる時間は増加し、探索速度が低下するという問題が生じる。

## 4　提案手法

提案手法では、直積量子化コードワードをベクトル量子化して距離表の計算を高速化することにより、マージした IVFPQ インデックスに対しても高速に探索を行えるようにする。以下具体的に述べる。なお、マージすべきインデックスの個数を $L$ とする。

$S$ 個の部分空間は独立であるから、一般性を失わず $s$ 番目 $(s \in \{1, 2, ..., S\})$ の部分空間について考える。いま、$s$ 番目の部分空間では、$L$ 個のインデックスそれぞれについて、$K$ 個の $D/S$ 次元直積量子化コードワードが存在する。この合計 $LK$ 個のコードワードを、まとめて $\mathcal{C}_s = \{\mathbf{c}^s_1, \mathbf{c}^s_2, \cdots, \mathbf{c}^s_{LK}\}$ と表記する。提案手法では、この $\mathcal{C}_s$ を $K$ 個のクラスタにクラスタリングし、新たなコードブック $\mathcal{C}'_s = \{\mathbf{c}'^s_1, \mathbf{c}'^s_2, \cdots, \mathbf{c}'^s_K\}$ を得る。そして、$\mathcal{C}_s$ のコードワードのそれぞれについて、$\mathcal{C}'_s$ のコードワードの中で最も近いものが何番目かを配列 $T_{near}$ に記録する。

次に、クエリ $\mathbf{q}$ に対する距離表の作成について、$s$ 番目のサブベクトルである $\mathbf{q}^s$ に注目して考える。提案手法では、$\mathbf{q}^s$ と、$\mathcal{C}'_s$ の $K$ 個のコードワードのみと距離計算を行い、距離表 $T_{dist}$ を作成する。そして、$\mathcal{C}_s$ のコードワードのうち $j$ 番目のコードワード $\mathbf{c}^s_j$ と $\mathbf{q}^s$ との距離は、$T_{near}$ を参照し、$\mathcal{C}'_s$ のコードワードの中で $\mathbf{c}^s_j$ に最も近いコードワードの $T_{dist}$ の値に近似する。

このように、提案手法では距離表の計算がテーブルのルックアップで近似的に行えるため、ベースラインよりも探索が高速になることが期待される。

## 5　実験と結果

提案手法の評価のために、SIFT1M, Deep1M の 2 つのデータセットを用いる (表 1)。今回の実験では、10 個の IVFPQ インデックスをマージする状況を想定する。

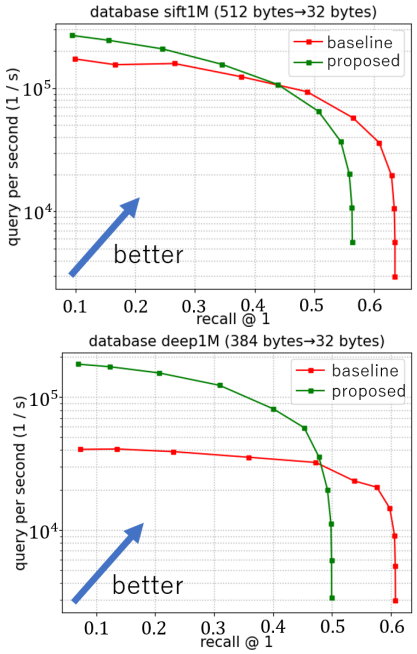


図 2　実験結果 $(K = 256, S = 32, R = \sqrt{100,000})$

表 1　実験で用いたデータセットの情報

| データセット | データ数 | クエリ数 | $D$ |
|---|---|---|---|
| SIFT1M | 1,000,000 | 10,000 | 128 |
| Deep1M | 1,000,000 | 10,000 | 96 |

そのために、各データセットについて、データベースベクトルの集合 $\mathcal{X}$ を $\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_{10}$ に 10 分割する。そして、10 個の IVFPQ インデックス $I_1, I_2, \cdots, I_{10}$ を独立に構築する。ここで、$I_m$ $(m \in \{1, 2, ..., 10\})$ はデータベースベクトルの集合として $\mathcal{X}_m$ を持ち、粗量子化、直積量子化のトレーニングも $\mathcal{X}_m$ で行う。

ベースラインは、3 章で述べたナイーブなマージ手法とする。一方提案手法は、ナイーブなマージ手法の距離表計算を、4 章で述べた近似計算で行う手法とする。この 2 つの手法の間で、精度-速度比を比較する。

実験結果を図 2 に示した。グラフは $p$ を動かし作成した。SIFT1M, Deep1M ともに、recall が比較的低い範囲では提案手法がベースラインよりも高速に探索を行えることが確認できた。これは、ベースラインのボトルネックであった距離表計算が、提案手法では高速化されているためだと考えられる。

## 参考文献

[1] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, Vol. 33, No. 1, p. 117–128, 2011.