大数据算法-2025 春

Lecture 10: Product Quantization

2025.4.1

Lecturer: 丁虎 *Scribe:* 王浩宇，莫官霖

1 Product Quantization(PQ) 内积量化 [3]

利用 k-means 作近似近邻查询，可以使用 $\{c_1, ..., c_k\}$ 来近似 P, 从 $\{c_1, ..., c_k\}$ 中找到 u 的最近邻。其查询时间为 $\Theta(kd)$. 极限情况 $k = n$，此时查询时间为 $\Theta(nd)$ 不过可以返回精确解。

如果我们将 R^d 分解为 $\mathrm{R}^{\frac{d}{m}} \times \mathrm{R}^{\frac{d}{m}} \times ...\mathrm{R}^{\frac{d}{m}}$ 对于每个 $\mathrm{R}^{\frac{d}{m}}$，都存在 P 到其上的投影 $P_j, 1 \leq j \leq m$. 我们对于 P_j 做 k-means, 也即找到 $\{c_1^j, ..., c_k^j\} \subset \mathrm{R}^{\frac{d}{m}}$ 将所有的中心建立一个"codebook" 如下

$$CB = \begin{pmatrix} c_1^1 & ... & c_k^1 \\ ... & ... & ... \\ c_m^1 & ... & c_m^k \end{pmatrix} \tag{1}$$

该 codebook 需要空间 $k \times m \times \frac{d}{m} = kd$.

之后我们建立表 $A \in Z^{m \times n}$, 其中第 (i, j) 个元素存储 P 中第 j 个点在第 i 个 $\mathrm{R}^{\frac{d}{m}}$ 子空间上的近似。如果 u 在这 m 个子空间中对应的中心分别为 $c_{t_1}^1, ..., c_{t_m}^m$, 那么对于 u, 我们存储列向量 $(t_1, ..., t_m)^\top$ 在 A 中。

之后建立表 $B \in \mathrm{R}^{m \times \binom{k}{2}}$, 其中第 i 行存储对应的子空间上 $c_1^i, ..., c_k^i$ 中心两两之间的距离。我们用 $B_{i,(s_i,t_i)}$ 表示在第 i 个子空间上 $c_{s_i}^i$ 与 $c_{t_i}^i$ 的距离。

我们的算法每次询问一个 $q \in \mathrm{R}^d$, 首先计算 q 与 CB 中类中心的距离，得到最近的中心对应的下标列向量 $S = (s_1, ..., s_m)^\top$，将其和 A 中每一列 $T = (t_1, ..., t_m)^\top$ 作比较，找到下标距离最近的列。这里的下标距离使用 B 中得到的对应距离。也即 $dist(S, T) = \sum_{i=1}^m B_{i,(s_i,t_i)}$. 这样的计算可以在 $\Theta(m)$ 时间完成。(这里我们简化了 A 和 B 的存储方式。) 输出下标距离最近的列对应的点。

这样操作的直觉在于 $\forall u, q \in P\ u = [u_1, ..., u_m]$ 以及 $q = [q_1, ..., q_m]$ 我们有

$$\begin{aligned}\|u - q\|^2 &= \sum_{i=1}^{m} \|u_i - q_i\|^2 \\ &\approx \sum_{i=1}^{m} \|c_{t_i}^i - c_{s_i}^i\|^2\end{aligned}$$

该算法复杂度分析

1. Construction Time $T(k-means) + \Theta(mn) + \Theta(k^2 m) = \Theta(knd) + \Theta(mn) + \Theta(k^2 m)$, 如果 $k.m \ll n, d$, 时间为 $\Theta(nd)$
2. Space $\Theta(mk\frac{d}{m}) + \Theta(mn) + \Theta(k^2 m) = \Theta(n + d)$.
3. Query Time $\Theta(mk\frac{d}{m}) + \Theta(m \times n) = \Theta(n + d) \ll nd$

当然实际使用中还有很多其他的问题，如何存储 A，B 以及如何优化 Codebook? 感兴趣的同学对这些后续问题可以自行了解。

本篇讲义主要聚焦于 [3] 的工作，这是 PQ 的开创性论文，首次系统提出了将高维向量空间分成子空间、分别进行量化的思想，显著减少存储和计算开销，同时保持良好的近似搜索精度。该方法已成为大规模图像检索的核心技术之一。在 PQ 被提出之后，[2] 提出了 Optimized Product Quantization (OPQ)，通过对原始向量进行旋转变换，使得每个子空间的量化误差更均匀，从而提升 PQ 的整体表示能力和搜索精度。进一步地，[1] 一文对 OPQ 做了更深入的理论分析和算法优化，实现了更高的准确率和速度平衡。该方法已广泛集成于 Facebook 的 FAISS 库中。

References

[1] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014.

[2] H. Jégou, M. Douze, and C. Schmid. Searching with quantization: approximate nearest neighbor search using short codes. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 728–735. IEEE, 2011.

[3] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.