大数据算法-2025 春

Lecture 10: Local Sensitive Hash

2024.4.11

Lecturer: 丁虎 *Scribe:* 王浩宇

这是一类近似近邻查询 (Approx-NN) 的方法，其主要思想是设计 Hash 函数，将点集 P 中的点放入不同的 Bucket 中，同时使得距离越近的点，其哈希值碰撞的概率越大。最后通过增加 hash 的次数来提升成功的概率。

1 近似近邻查询

Definition 1.1 ((r, R)-近似近邻查询)**.** 输入集合 $P \subset \mathrm{R}^d$ 和一个点 $q \in \mathrm{R}^d$, 我们记 $dist(q, p) = min_{p \in P} \|q - p\|$。$(r, R)$-近似近邻查询要求

1. 如果 $dist(q, P) \leq r$, 返回 $u \in P$ 使得 $\|q - u\| \leq R$.
2. 如果 $dist(q, P) > R$, 输出“$dist(q, P) > r$”.
3. 如果 $r < dist(q, P) \leq R$, 返回上面两者中任意一种。

如果 $R = r$ 那即为精确的 $NS(r)$. 通常我们会考虑 $R = (1 + \epsilon)r$ 的情况。这个时候我们可以称上面的问题为 $(1 + \epsilon)$-近似近邻查询。

在已知 $dist(q, P)$ 的上下界为 $[a, b]$ 的情况下，建立集合 $U = [a, (1+\epsilon)a, ..., (1+\epsilon)^{\log_{1+\epsilon} \frac{b}{a}} a]$，然后对于 r 的选择，我们在 U 上作 Binary Search，这样我们可以对通过至多 $\log_{1+\epsilon} \frac{b}{a}$ 次 NS(近邻查询) 来实现 $(1 + \epsilon)$-近似近邻查询。

我们用 $B(q, r)$ 表示以 q 为球心，r 为半径的欧几里得距离的球体。用 $N(\vec{0}, I_d)$ 表示 d 维标准正态分布。$U([a, b])$ 表示区间 $[a, b]$ 上的均匀随机分布。

Definition 1.2. (r, R, α, β)-Sensitive Hash 给定 $r < R, 1 > \alpha > \beta > 0$, 我们称 F 是一个 (r, R, α, β)-Sensitive Hash 映射集合，如果对于 $\forall u, q \in \mathrm{R}^d$, 随机取 $h \in F$ 满足

1. 如果 $u \in B(q, r)$, 那么 $\Pr[h(u) = h(q)] \geq \alpha$.
2. 如果 $u \notin B(q, R)$, 那么 $\Pr[h(u) = h(q)] \leq \beta$.

下面我们给一个 F 的构造方法。考虑这样的 $F_T = \{h|h(u) = \lfloor\frac{\langle u,\vec{v_h}\rangle+t_h}{T}\rfloor\}$, 其中 T 是一个待确定的值，$\vec{v_h} \sim N(\vec{0}, I_d)$,$t_h \sim U([0,T])$. 我们有如下定理

Theorem 1.3. *任给 $r, \epsilon > 0$,$1 > \alpha > \beta > 0$, 且 $\frac{\log\frac{1}{\alpha}}{\log\frac{1}{\beta}} \le \frac{1}{1+\epsilon}$, 则一定存在 $T > 0$, 使得 F_T 是 $(r, (1+\epsilon)r, \alpha, \beta)-$Sensitive 的。*

注意到上面的 h 是将 $\mathrm{R}^d \to \mathrm{Z}$, 下面我们讨论如果将该 Hash 过程的成功率提高。为此我们考虑一个新的哈希函数 $g : \mathrm{R}^d \to \mathrm{Z}^k$, 其中 $g = (h_1, h_2, ..., h_k)$,$h_i \in F$. 这样构成的函数族我们称为 $G(F, k) = g = (h_1, ..., h_k|h_i \in F)$.

那么我们有下面的结论

Theorem 1.4. *令 $\rho = \frac{\log\frac{1}{\alpha}}{\log\frac{1}{\beta}} \le \frac{1}{1+\epsilon}$, $k = \log_{\frac{1}{\beta}} n$, $\tau = 2n^\rho = o(n)$. 随机从 $G(F, k)$ 中取出 $g_1, g_2, ..., g_\tau$ 其对应 τ 个哈希 bucket $H_1, ..., H_\tau$, 我们有对于 $\forall q \in \mathrm{R}^d$, 满足下列两个条件的概率 $\ge \frac{3}{5}$:*

1. *如果 $\exists u \in P$, 使得 $\|u - q\| \le r$, 则 $\exists j$ 使得 $g_j(u) = g_j(q)$.*
2. *如果 $\forall u \in P$, $\|u - q\| > (1 + \epsilon)r$, 则在 $H_1, H_2, ..., H_\tau$ 中与 q 发生冲突的点 $\le 4\tau$.*

Proof. 先证明 2。

$$\begin{aligned}
&\forall u \in P, \|u - q\| > (1 + \epsilon)r \\
&\Rightarrow \forall g \in G(F, k), \Pr[g(u) = g(q)] \le \beta^k = \frac{1}{n} \\
&\Rightarrow \forall H_j, \mathrm{E}[\text{发生冲突个数}] \le 1 \\
&\Rightarrow \text{在}H_1 \sim H_\tau\text{中发生冲突总数期望} \le \tau \\
&\Rightarrow \Pr[\text{发生冲突个数} \le 4\tau] \ge \frac{3}{4}
\end{aligned}$$

再证明 1.

$$\begin{aligned}
&\|u - q\| \le r \\
&\Rightarrow \forall q, \Pr[g(u) = g(q)] \ge \alpha^k = n^{-\rho} \\
&\Rightarrow H_1 \sim H_\tau\text{至少发生一次冲突的概率} \ge 1 - (1 - n^{-\rho})^\tau = 1 - (1 - n^{-\rho})^{2n^\rho} \ge 1 - \frac{1}{\epsilon^2} > \frac{4}{5}.
\end{aligned}$$

□

由该定理，我们可以将 P 中的点分为 3 个类:

1. $\|u-q\| \leq r$, 此时发生冲突个数 ≥ 1

2. $\|u-q\| > (1+\epsilon)r$, 此时发生冲突个数 $\leq 4\tau$

3. $r < \|u-q\| \leq (1+\epsilon)r$, 此时发生冲突个数可能很多也可能很少，所以返回任意情况

这样我们只需要检查前面 $\leq 4\tau+1$ 个冲突即可。

该算法复杂的分析:

1. Construction Time $O(\tau \cdot n \cdot k \cdot d) = O(n^{1+\frac{1}{1+\epsilon}} d \log n) \leq O(n^2 d)$.

2. Space $O(nd + \tau \cdot k \cdot n) = O(nd + n^{1+\frac{1}{1+\epsilon}} \log n)$.

3. Query Time $O(\tau \cdot k \cdot d + (4\tau+1)d) = O(n\frac{1}{1+\epsilon} \log n \cdot d) < nd$.

2 Product Quantization(PQ) 内积量化

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

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

该算法复杂度分析

1. Construction Time $T(k-means) + \Theta(mn) + \Theta(k^2m) = \Theta(knd) + \Theta(mn) + \Theta(k^2m)$, 如果 $k.m \ll n, d$, 时间为 $\Theta(nd)$

2. Space $\Theta(mk\frac{d}{m}) + \Theta(mn) + \Theta(k^2m) = \Theta(n+d)$.

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。