局部敏感哈希（LSH）

秦睿哲

一、概述

LSH(Locality Sensitive Hashing)，中文叫做“局部敏感哈希”，它是一种针对海量高维数据的快速最近邻查找算法。在信息检索、数据挖掘以及推荐系统等应用中，我们经常会遇到的一个问题就是面临着海量的高维数据，查找最近邻。如果使用线性查找，那么对于低维数据效率尚可，而对于高维数据，就显得非常耗时了。为了解决这样的问题，人们设计了一种特殊的 hash 函数，使得 2 个相似度很高的数据以较高的概率映射成同一个 hash 值，而令 2 个相似度很低的数据以极低的概率映射成同一个 hash 值。我们把这样的函数，叫做 LSH（局部敏感哈希）。LSH 最根本的作用，就是能高效处理海量高维数据的最近邻问题。

二、定义

LSH Family: 对于一个相似性函数 s，局部敏感哈希族 H 是一组哈希函数，对于任意两个对象 $p,q \in B$ 有

$$\Pr_{h\in H}\left[h(p)=h(q)\right]\approx s(p,q)$$

即对于任意的 $p,q \in B$，一个随机选择的哈希函数 $h \in H$ 会导致这两个对象映射为同一哈希值的概率与他们本身的相似度近似。就是当足够相似时，映射为同一 hash 值的概率足够大；而足够不相似时，映射为同一 hash 值的概率足够小。

LSH：$r < R, 1 > \alpha > \beta > 0$，$H = \{h\}$是一个$(r, R, \alpha, \beta)-$sensitive哈希映射集合，如果：对于 $\forall u,q \in R^d$ 随机取 $h \in H$，则

(1) 如果$u \in \text{Ball}(q,r), \text{Prob}[h(u)=h(q)] \geq \alpha$

(2) 如果$u \notin \text{Ball}(q,R), \text{Prob}[h(u)=h(q)] \leq \beta$

其中 u,q 表示两个具有多维属性的数据对象，$u \in \text{Ball}(q,r)$或者$u \notin \text{Ball}(q,R)$表示了 2 个对象的相异程度。相似度的定义根据实际情况自己决定，针对不同的相似度测量方法，局部敏感哈希的算法设计也不同。无论是哪种算法，都是将高维数据降维到低维数据，同时，还能在一定程度上，保持原始数据的相似度不变。LSH 不是确定性的，而是概率性的，也就是说有一定的概率导致原本很相似的数据映射成 2 个不同的 hash 值，或者原本不相似的数据映射成同一 hash 值。这是高维数据降维过程中所不能避免的（因为降维势必会造成某种程度上数据的失真），不过好在 LSH 的设计能够通过相应的参数控制出现这种错误的概率，这

也是 LSH 被广泛应用的原因。

三、构造

课上提出了一种 LSH 函数的构造方法：

$$h(u)=\left\lfloor \frac{\langle u,\vec{v}_h\rangle+t_h}{T} \right\rfloor$$

(1) T 是一个固定的值，适用于所有 H 中的 h；

(2) $\vec{v}_h \sim N^d(0,1)$；

(3) $t_h \in [0,T]$ 中随机选取的一个值。

四、LSH 的建造

1、一个结论：

任给 $r,\varepsilon>0$，$1>\alpha>\beta>0$ 且 $\dfrac{\log\frac{1}{\alpha}}{\log\frac{1}{\beta}} \le \dfrac{1}{1+\varepsilon}$，则一定存在 $T>0$，使得 H_T 是 $(r,(1+\varepsilon)r,\alpha,\beta)-\text{sensitive}$的。

注意到，$\dfrac{\log\frac{1}{\alpha}}{\log\frac{1}{\beta}} \le \dfrac{1}{1+\varepsilon}$ 就等价于 α 和 β 之间有一定的距离，这个距离由参数 ε 决定。

2、哈希表

为了减少漏报率（就是本来很相近的两条数据被认为是不相似的），我们的解决方案是用多个哈希函数对向量执行 hash 运算。比如说，对任意一个向量 v，现在准备了 k 个哈希函数 $h_1,h_2,\cdots,h_k$，这 k 个哈希函数是我们通过构造得到的随机 k 个。通过计算，就得到了 k 个 hash 值。我们记 $g=(h_1,h_2,\cdots,h_k)$。由于每个哈希函数 $h:R^d \to Z$，所以 $g:R^d \to Z^k$。最后，我们令这个整体的哈希为

$$G(H,k)=\{g=(h_1,h_2,\cdots,h_k)\,|\,h_i \in H\}$$

3、建造过程

要求：H 是 $(r,(1+\varepsilon)r,\alpha,\beta)-\text{sensitive}$的，$n=|P|$ 是点集的个数，$\rho=\dfrac{\log\frac{1}{\alpha}}{\log\frac{1}{\beta}} \le \dfrac{1}{1+\varepsilon}$，

$k=\log_{\frac{1}{\beta}} n$， $\tau=2n^{\rho}=o(n)$ [1]

我们随机地从$G(H,k)$中选取τ个哈希映射$g_1,g_2,\cdots,g_\tau$。现在这τ个函数分别对数据处理，只要有一组完全相等，就认为两条数据是相近的。我们可以对这种方法进行简单的分析：

一个简单的例子，我们通过第三部分的构造方法得到一个哈希族$H=\{h\}$，因此我们有$\Pr_{h\in H}[h(p)=h(q)]\approx s(p,q)$。简单起见，记$s=s(p,q)$，我们将分析$\tau$个$k$维哈希函数的情况[2]：

$$
\begin{aligned}
s &= \text{任何一个哈希函数相等的概率}\\
s^k &= g\text{中所有的哈希函数都相等的概率}\\
(1-s^k) &= g\text{中不是所有的哈希函数都相等的概率}\\
(1-s^k)^\tau &= \text{没有一个}g_i\text{，满足所有的哈希函数都相等的概率}\\
1-(1-s^k)^\tau &= \text{至少有一个}g_i\text{中的所有的哈希函数都相等的概率}
\end{aligned}
$$

最后的公式给出了，使用τ个哈希映射$g_1,g_2,\cdots,g_\tau$，每个g_i使用k个哈希函数的情况下，查找到最近邻的概率$P=1-(1-s^k)^\tau$。

4、补充

引理：任给一个查询点$q\in R^d$，满足下列两个条件的概率$\geq\frac{3}{5}$

(1) 如果$\exists u\in P,\|u-q\|\leq r,$则存在$j,\ s.t.\ g_j(u)=g_j(q)$

(2) 如果$\forall u\in P,\|u-q\|>(1+\varepsilon)r,$则在哈希表$H_1\sim H_\tau$中（理解为$g_1,g_2,\cdots,g_\tau$的值域），与$q$发生冲突的点的个数$\leq 4\tau$。

证明：我们首先考虑(2)。

$$\forall u\in P,\|u-q\|>(1+\varepsilon)r$$

$$\Rightarrow \forall g\in G(H,k),\ \mathrm{Prob}[\mathrm{g(u)}=g(q)]\leq\beta^k=\frac{1}{n}$$

$$\Rightarrow \forall H_j\ \ E[\text{发生冲突的点的个数}]\leq 1$$

$$\Rightarrow \text{在}H_1\sim H_\tau\text{中发生冲突的总数的期望}\leq\tau$$

$$\Rightarrow \text{发生冲突总数}\leq 4\tau\text{的概率}\geq\frac{3}{4}\qquad\text{（利用切比雪夫不等式）}$$

再考虑(1) $\|u-q\| \le r$

$\Rightarrow \forall g, \text{Prob}[g(u)=g(q)] \ge 2^k = n^{-\rho}$

$\Rightarrow H_1 \sim H_\tau$ 中至少发生一次冲突的概率 $\ge 1-(1-n^{-\rho})^\tau = 1-(1-n^{-\rho})^{2n^\rho} \ge 1-\frac{1}{e^2} > \frac{4}{5}$

把 P 中的点分成 3 种类型：①到 q 的距离 $\le r$ ②到 q 的距离 $> (1+\varepsilon)r$ ③ $r <$ 到 q 的距离 $\le (1+\varepsilon)r$

① 发生冲突的总数 ≥ 1

② 发生冲突的总数 $\le 4\tau$

③ 发生冲突的总数可能很多，也可能很少

因此我们只须查看前面 $4\tau+1$ 个冲突即可。作近邻查询：

① $\exists u \in P, \|u-q\| \le r$, 返回一个点 u', $s.t. \|u'-q\| \le (1+\varepsilon)r$

② $\forall u \in P, \|u-q\| > (1+\varepsilon)r$, 返回"所有点离 q 的距离 $> r$"

③ $r < \text{dist}(q,p) \le (1+\varepsilon)r$, 两种返回都可以

5、查询问题的复杂度

① 时间（建立 Hash）：$O(\tau \cdot n \cdot kd) = O\left(n^{1+\frac{1}{1+\varepsilon}} d \cdot \log n\right)$

② 空间：$O(nd + \tau \cdot kn) = O\left(nd + n^{1+\frac{1}{1+\varepsilon}} \log n\right)$

③ 查询时间：

$$
\begin{aligned}
& O(\tau \cdot kd + (4\tau+1)d) \\
&= O\left(n^{\frac{1}{1+\varepsilon}} \log n \cdot d\right) \\
&= o(nd)
\end{aligned}
$$

参考文献：

[1] Andoni A , Indyk P . *Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions*[C]// 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings. IEEE, 2006.

[2] Jeff M. Phillips. *MATHEMATICAL FOUNDATIONS FOR DATA ANALYSIS.* Section 4.6 Locality Sensitive Hashing.