

程序设计实习（实验班-2025春）

距离度量及其计算

授课教师：姜少峰

助教：管晏如、孔启皓、楼家宁、杨卓凡

Email: shaofeng.jiang@pku.edu.cn

距离与相似度

- 给定一个数据集 X , 定义一个距离/相似度度量 $d : X \times X \rightarrow \mathbb{R}_+$

- 相似度: 越大越相似, 越小越不相似

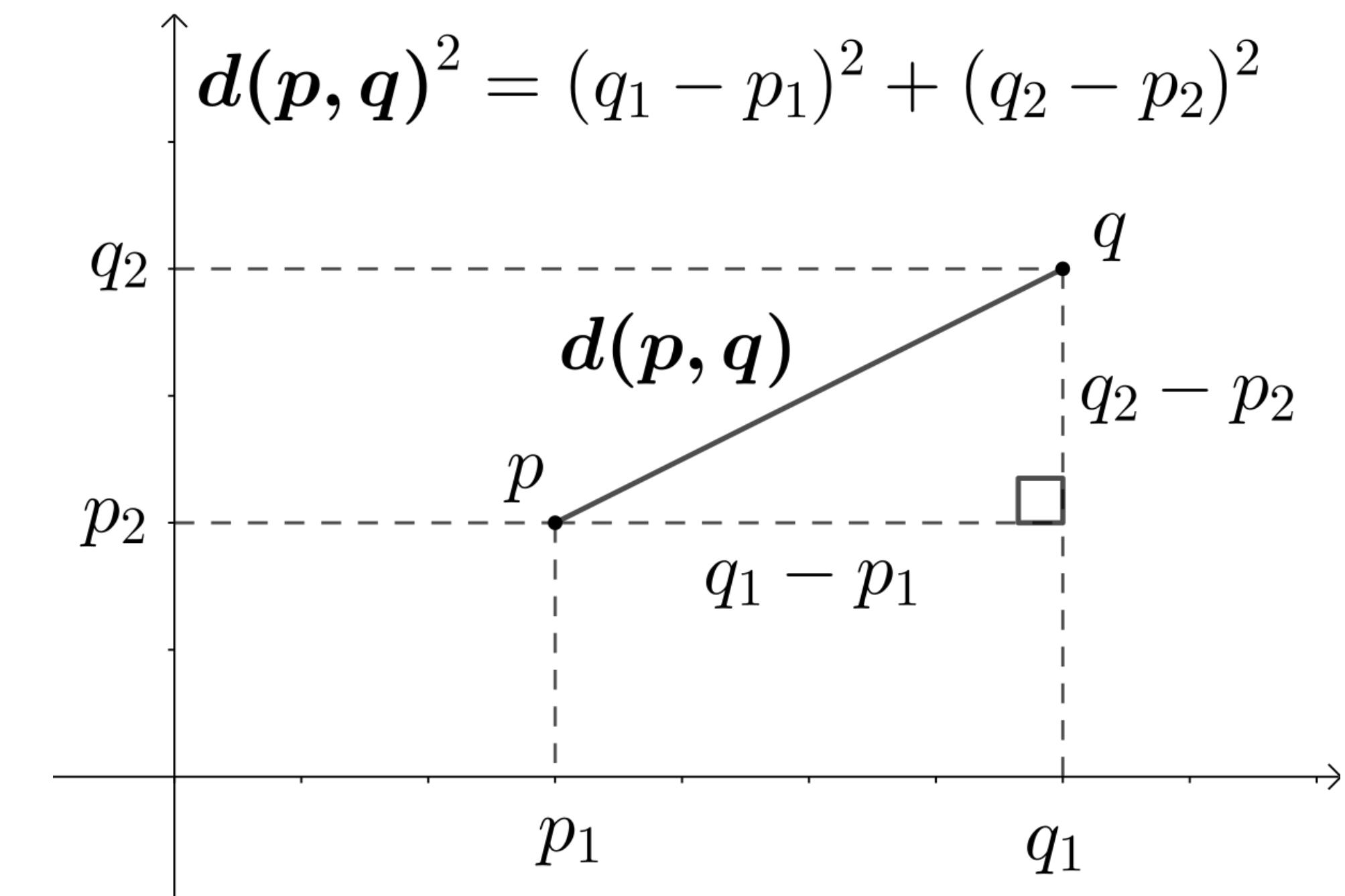
最常见的是距离: 欧氏距离

- 距离: “不相似度”

- 越小越相似, 越大越不相似

- 一般满足三角形不等式

$$d(x, y) + d(y, z) \geq d(x, z)$$



更多解释

- 相似度不满足三角形不等式的解释：
 - 考虑三个元素 a b c , a b 很相似, b c 很相似, 那么很自然的有 a c 很相似
 - 也就是 $\text{sim}(a, b)$ 和 $\text{sim}(b, c)$ 很大, 并且可能 $\text{sim}(a, c)$ 比二者的和还大
- 不相似度/距离 满足 三角形不等式的解释：
 - 考虑反面、如果不满足
 - 那么 a b 距离不大且 b c 距离不大时, a c 却可能距离远, 这是不合理的

基于相似度/距离的应用举例

推荐系统

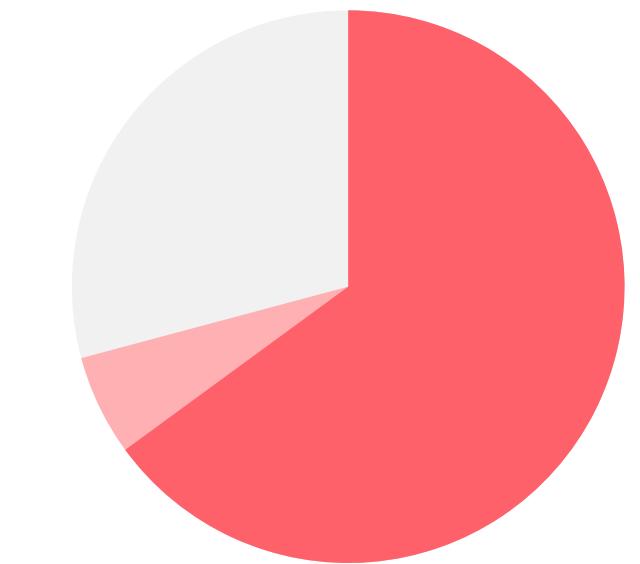
	Item 1	Item 2	Item 3	Item 4	Item 5
Alice		👎		👍	
Bob	👍	👍		👎	👍
Charlie	👍		👎	?	👍

Bob ~ Charlie → ? = 👎

抄袭检测

类似的：网页去重

29
RESULTS FOUND



- Identical
- Minor Changes
- Omitted Words

73%
MATCH

 See Alerts

There are billions of galaxies filled with billions of stars. Each star has the potential to have planets orbiting it. Does life exist on some of those planets? In this module, students discover how scientists find planets and other astronomical bodies through the wobble (also known as Doppler spectroscopy or radial-velocity) and transit methods. Students compare zones of habitability around different star types, discovering the zone of liquid water possibility around each star type. Finally, students explore how scientists use spectroscopy to learn about atmospheres on distant planets.

first is to study the effects of exposure to microgravity on biological systems to reduce the risks of manned space flight. The second is to use the microgravity environment to broaden scientific knowledge about the influence of gravity on living systems.

RESULTS (20)

Galaxy | Wikipedia.org

<https://en.wikipedia.org/wiki/Galaxy>

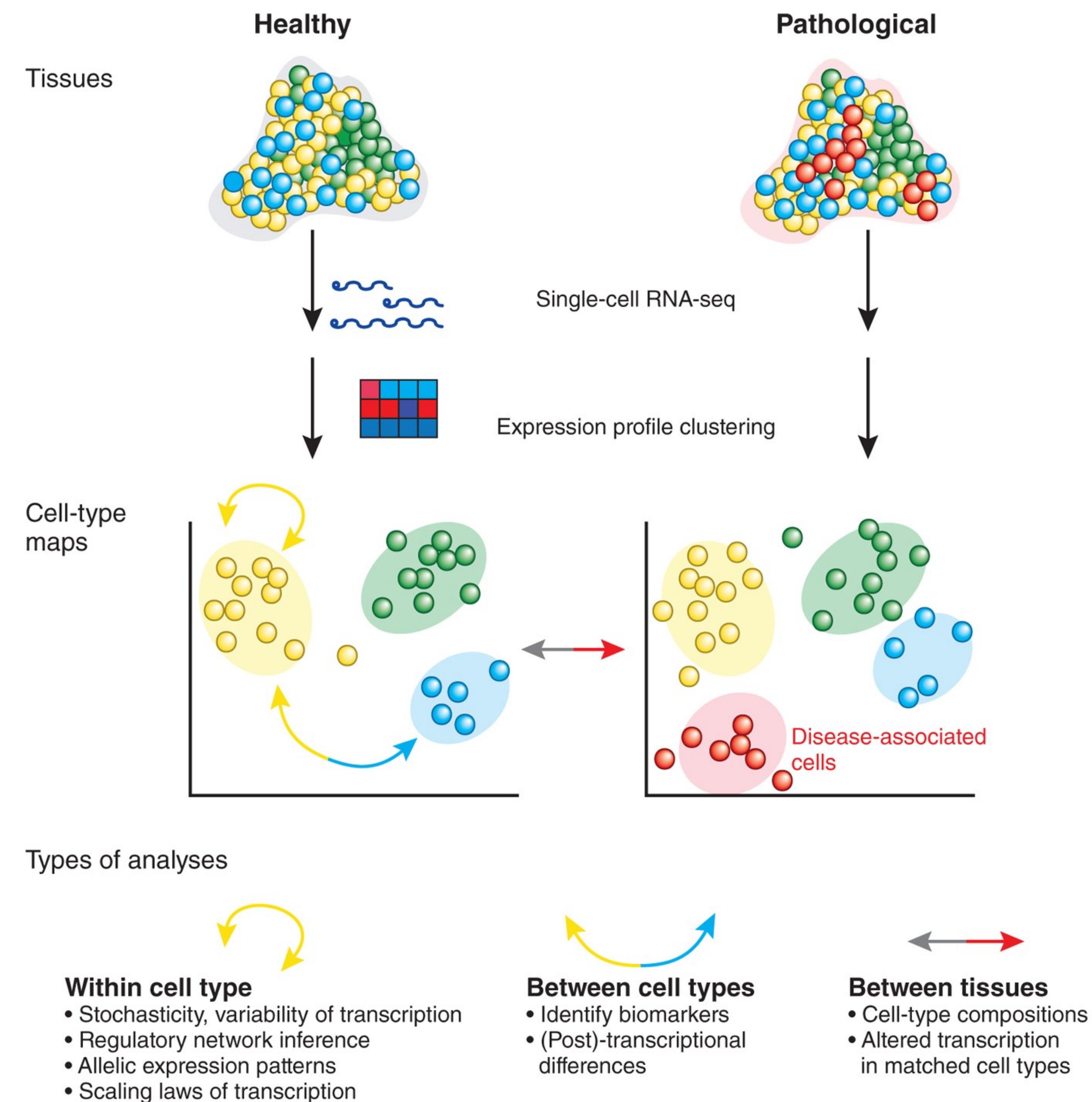
Skip to content...

46% Similar words

Space - High Adventure Science:
Is there lie in space? |
Wikipedia.org

<https://en.wikipedia.org/wiki/Space-High-a...>

生物学应用



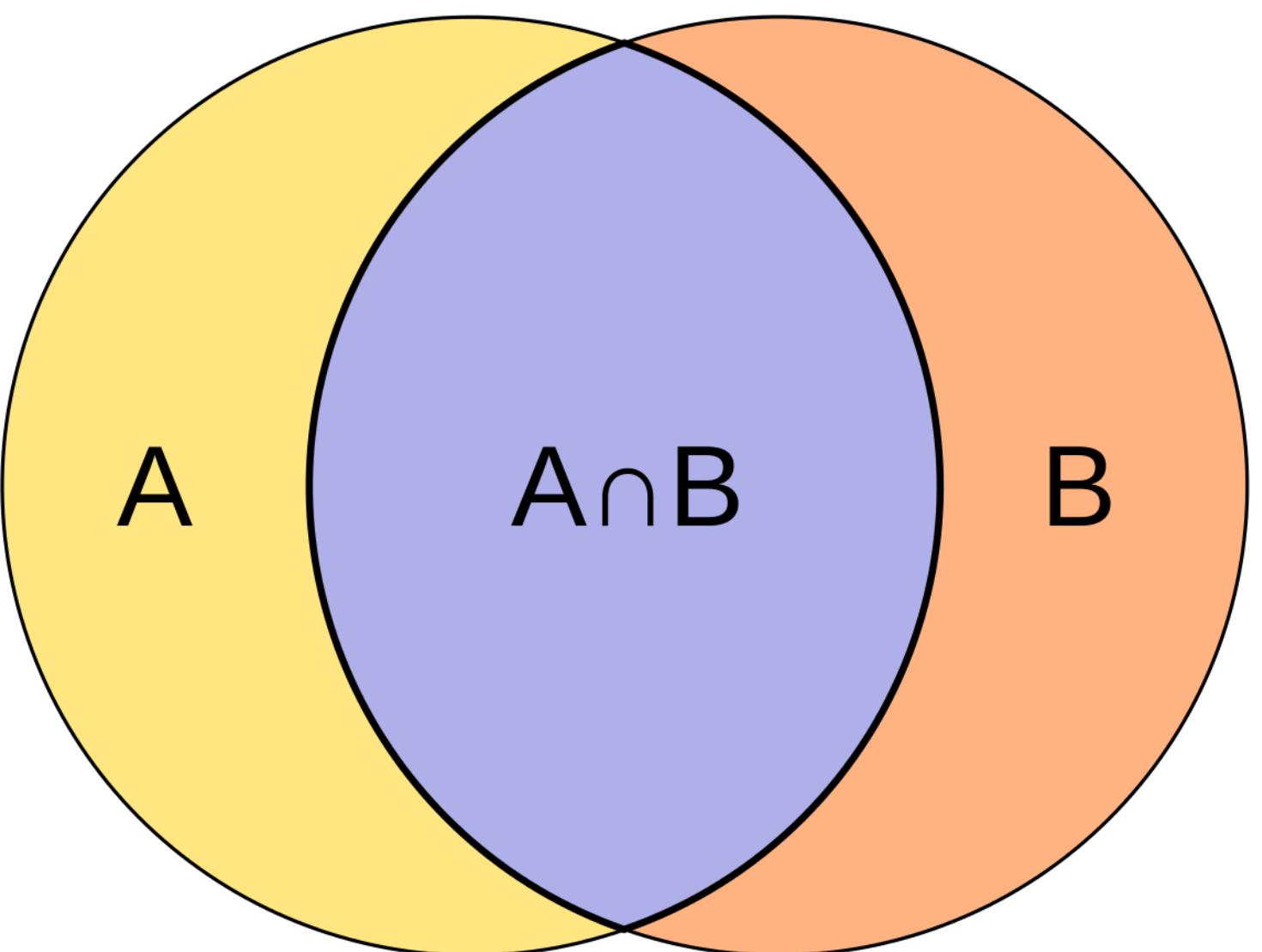
集合的相似度度量

Jaccard Similarity

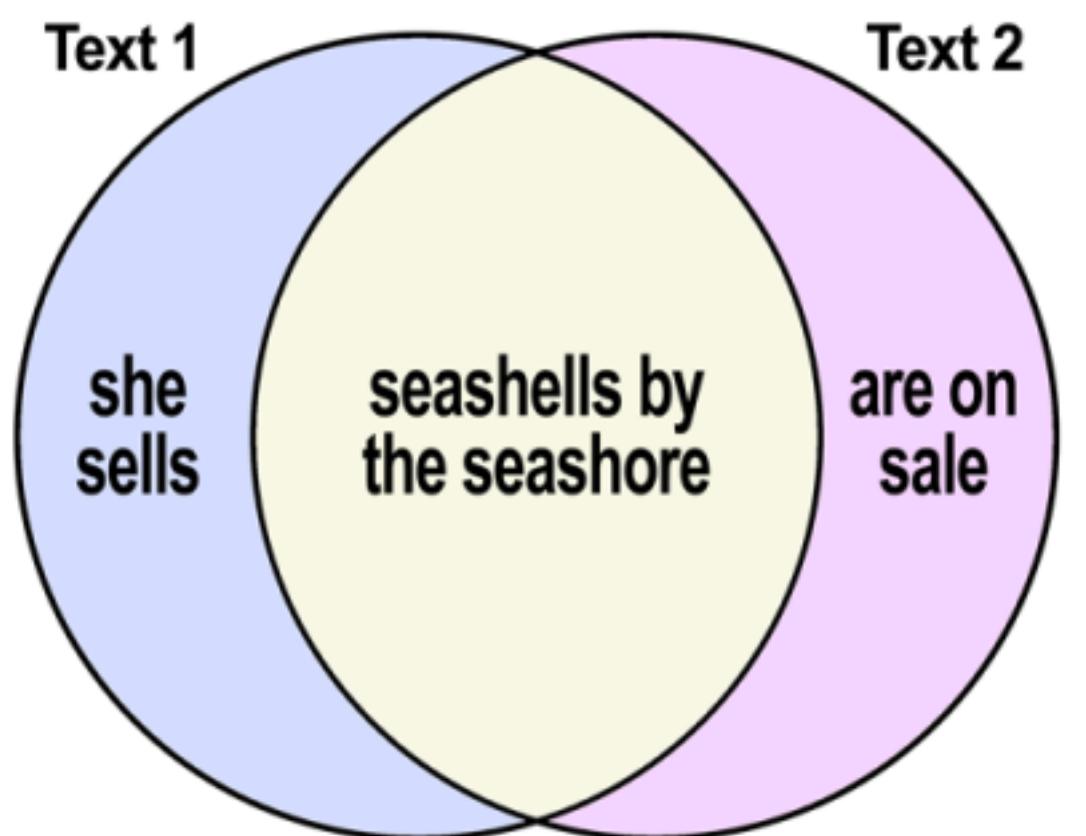
- 两个集合 $A, B \subseteq [n]$

是一个[0,1]的数

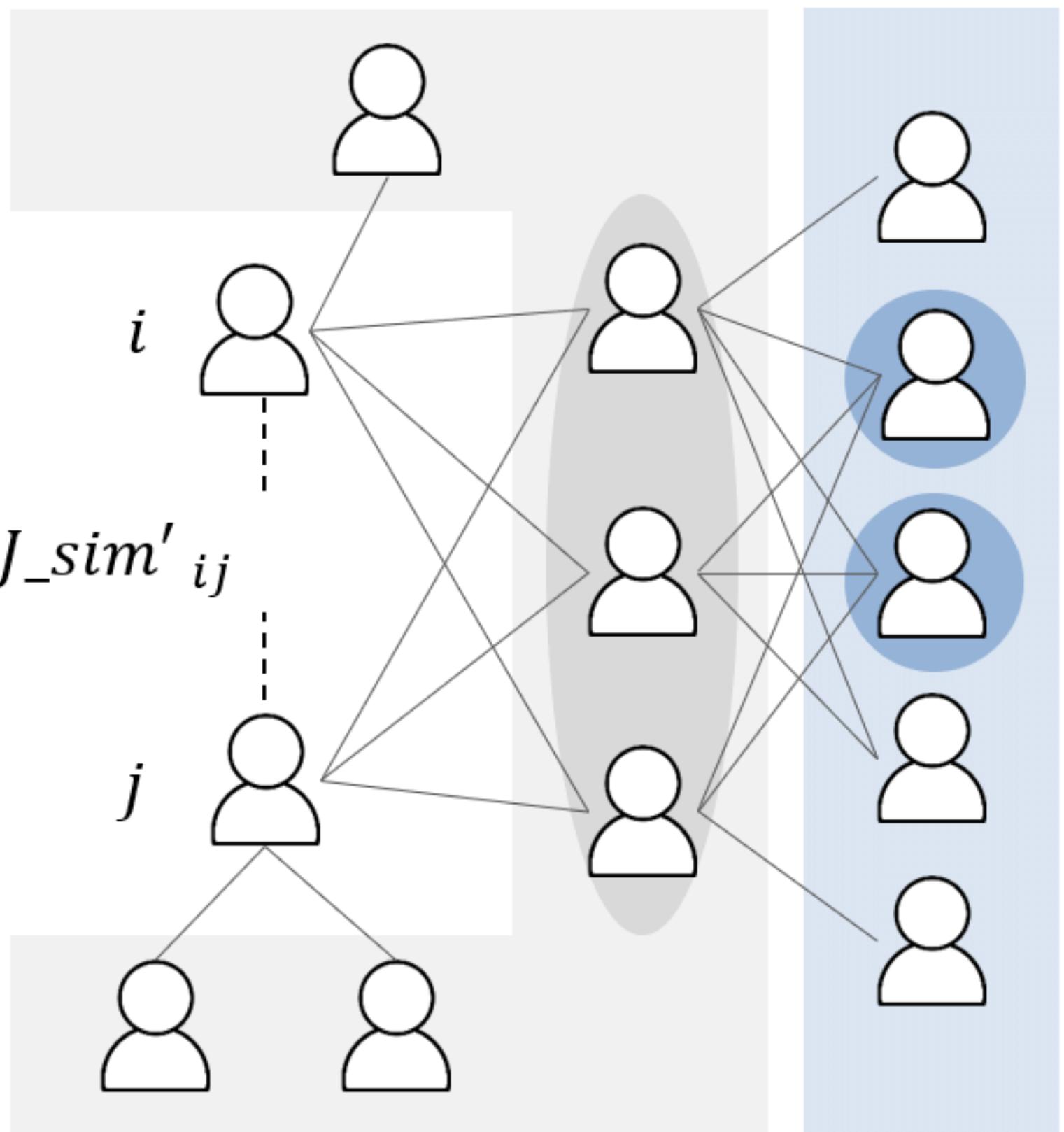
$$J(A, B) := \begin{cases} \frac{|A \cap B|}{|A \cup B|} & A \cup B \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$



文本的相似度（考虑单词的集合）



社交网络上用户的社交距离（考虑各自邻居的集合）



如何计算 $J(A, B)$?

$$J(A, B) := \begin{cases} \frac{|A \cap B|}{|A \cup B|} & A \cup B \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

- 直接计算：借助集合数据结构（设 $O(1)$ 代价），可以在 $O(|A| + |B|)$ 计算
- 如果数据集有很多点（集合） $\{A_i\}_i$ ，经常需要查询两个点的 $J(A_i, A_j)$ 呢？

一种基于哈希的思路

- 对每个集合 A_i , 希望计算/预处理一个哈希值 $h(A_i)$
- 使得 $J(A_i, A_j)$ 可以通过 $h(A_i)$ 和 $h(A_j)$ 直接得到
- 只需要 $O(|A_i|)$ 时间预处理 $h(A_i)$, 之后查询 $J(A_i, A_j)$ 都是 $O(1)$

MinHash

- 设所有的集合 A_i 的元素都来自某个universe $[n]$
- 设 $h : [n] \rightarrow [0,1]$ 为一个随机哈希 注意此处映射到实数
- 对每个 A_i , 计算 $h_{\min}(A_i) := \min_{x \in A_i} h(x)$

Claim: $\Pr[h_{\min}(A) = h_{\min}(B)] = J(A, B)$

h_{\min} 的性质

- 一般地，考虑一个任意的 $S \subseteq [n]$ ，并考察 $h_{\min}(S) := \min_{x \in S} h(x)$
- 由于所有 x 上的 $h(x)$ 是独立同分布的，任何 $x \in S$ 成为 \min 的概率都是一样的
- 更一般的，如果把 S 元素按照 $h(x)$ 升序排列，则会产生一个（均匀）随机排列
- 因此 MinHash 等价于把 $[n]$ 上的元素做随机排列

直接存随机排列是低效的
hash可在用到某value才现算现存

验证MinHash主要性质

主要思路：考慮 $h_{\min}(A \cup B)$

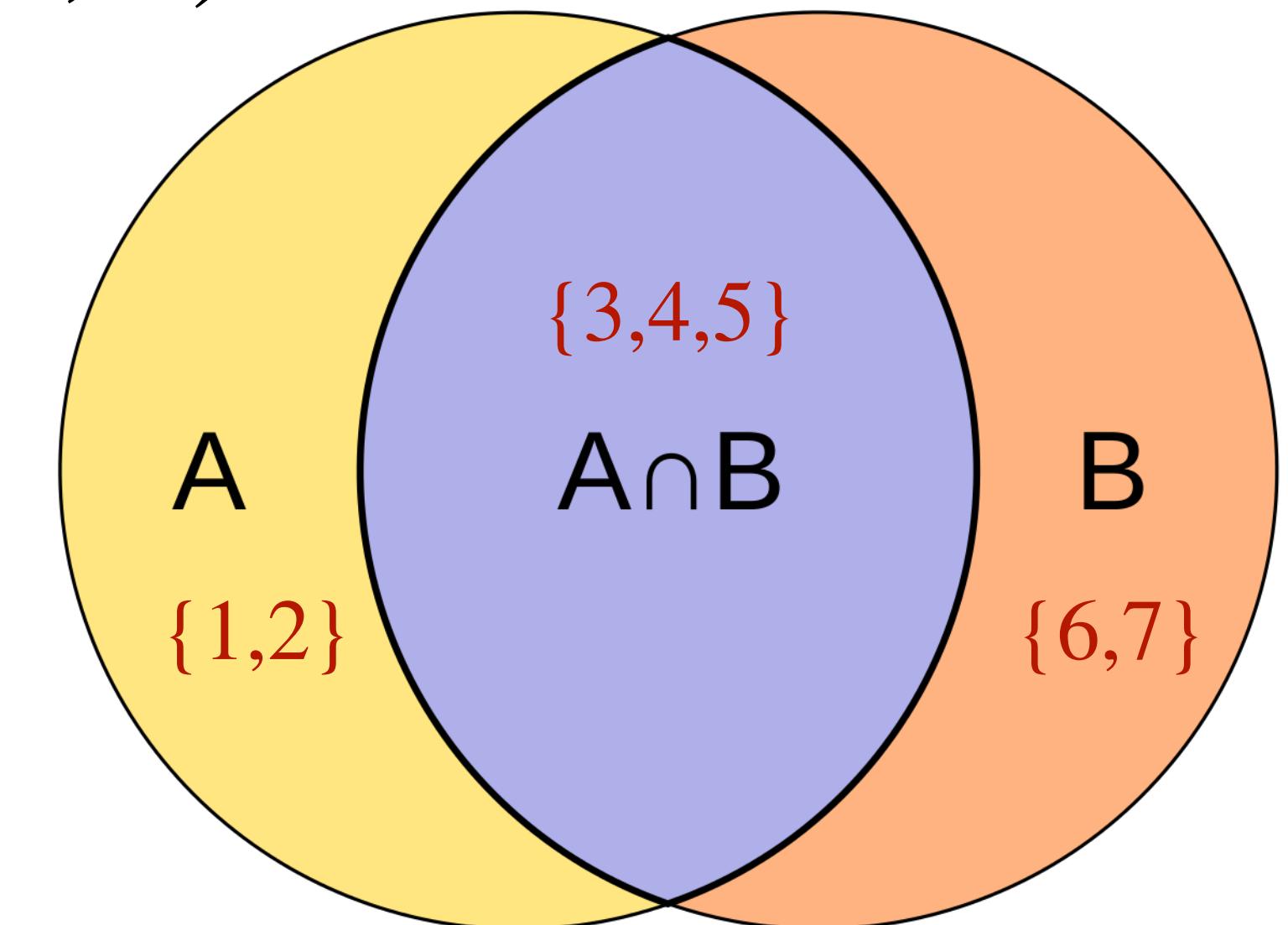
Claim: $\Pr[h_{\min}(A) = h_{\min}(B)] = J(A, B)$

- 考慮 $A = \{1,2,3,4,5\}$ 和 $B = \{3,4,5,6,7\}$ 两个集合
- 可以假定 h 在 $\{1, \dots, 7\}$ 上各不相等
- 观察: $h_{\min}(A) = h_{\min}(B)$ 仅当 $h_{\min}(A \cup B)$ 取在 $A \cap B$
 - 例如, $h_{\min}(B)$ 只能取在 $\{3 \dots 7\}$ 上, 若 $h_{\min}(A)$ 取在 $\{1,2\}$ 上则一定不能相等
 - 一共7个元素, min取在中间3个元素上, 概率是 $3/7$

$|A \cup B|$

$|A \cap B|$

$|A \cap B| / |A \cup B| = J(A, B)$



转化成随机算法

- 设 $X := \begin{cases} 1 & h_{\min}(A_i) = h_{\min}(A_j), \\ 0 & \text{oth.} \end{cases}$, 则 $\Pr[X = 1] = J(A_i, A_j)$
 - 完整算法（多次试验取平均值）：
 - 采用 T 个独立的随机哈希 $h^{(1)}, \dots, h^{(T)}$
 - 对每个 A_i 和 $h^{(t)}$, 计算 $h_{\min}^{(t)}(A_i)$
 - 对查询 A_i, A_j , 计算 $\frac{1}{T} \cdot |\{t \in [T] : h_{\min}^{(t)}(A_i) = h_{\min}^{(t)}(A_j)\}|$
- 如何实现?
- $O(|A_i| \cdot T)$ 时间
- 计算的是 $h_{\min}^{(t)}$ 相等的 t 所占比例
- $O(|T|)$ 时间

T取多大?

- 设估计量 $X := \frac{1}{T} \cdot |\{t \in [T] : h_{\min}^{(t)}(A_i) = h_{\min}^{(t)}(A_j)\}|$

Chernoff bound. 设 $X_1, \dots, X_n \in [0,1]$ 是独立、同期望 $\mu \geq t$ 随机变量

令 $X := (X_1 + \dots + X_n)/n$ 。对任何失败概率 $\delta \in (0,1)$, 有

$$\Pr \left[|X - \mu| \geq \sqrt{\frac{\log(1/\delta)}{nt}} \mu \right] \leq \delta$$

- 设 $X_t \in \{0,1\}$, such that $X_t = 1 \iff h_{\min}^{(t)}(A_i) = h_{\min}^{(t)}(A_j)$

思考: $J(A_i, A_j) \rightarrow 0$ 会怎样?

Chernoff告诉我们 $T = \frac{\log(1/\delta)}{\epsilon^2 \cdot J(A_i, A_j)}$ 时有失败概率 δ , 相对误差 ϵ

如何用小空间实现映射到 $[0, 1]$ 的哈希

- 考虑随机 $h : [n] \rightarrow [m]$, 用巨大的 m 以及 h/m 的值来模拟 $[0, 1]$ 均匀分布
- Universal hashing考虑的是 $m < n$ 的情况, 我们要利用它得到大 m 的hash
- 构造 $T := O(\log m)$ 个独立的 $\{h_i : [n] \rightarrow \{0, 1\}\}_{i=1}^T$ 的universal hashing
- 然后拼接起来形成一个新的 h , 即 $h(x) := (h_1(x), \dots, h_T(x))$
- 该函数 h 使用 $O(\log m \log n)$ 个随机数映射到一般的大小为 m 的域

作业八：用MinHash近似Jaccard Similarity

- <http://cssyb.openjudge.cn/25hw8/>
- Deadline: 4月2日
- 分值2分

向量的相似性度量

Cosine Similarity

- 考虑两个向量 $x, y \in \mathbb{R}^d$, 定义 $\sigma(x, y) := \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2}$ $\sigma(x, y) = \cos(\theta(x, y))$
- 典型应用：文本相似度 (TF-IDF)
 - 考虑一组文档 D , 要把某个文档 $d \in D$ 表示成一个向量
 - TF = term frequency: 一个单词在给定文档 d 中出现的频率
 - IDF = inverse document freq.: $\log(\text{总文档数} / \text{单词出现在多少文档})$
 - 对单词 w , $\text{TF-IDF}(w, d) = \text{TF}(w, d) * \text{IDF}(w, D)$
 - 对于一个文档 $d \in D$, 做成一个下标是单词 w 、值是 $\text{TF-IDF}(w, d)$ 的向量

一般非常的高维、稀疏，可以用 cosine similarity 衡量相似度

IDF的意义

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

We see that "[Romeo](#)", "[Falstaff](#)", and "salad" appears in very few plays, so seeing these words, one could be quite certain which play it is. In contrast, "good" and "sweet" appears in every play and are completely uninformative as to which play it is.

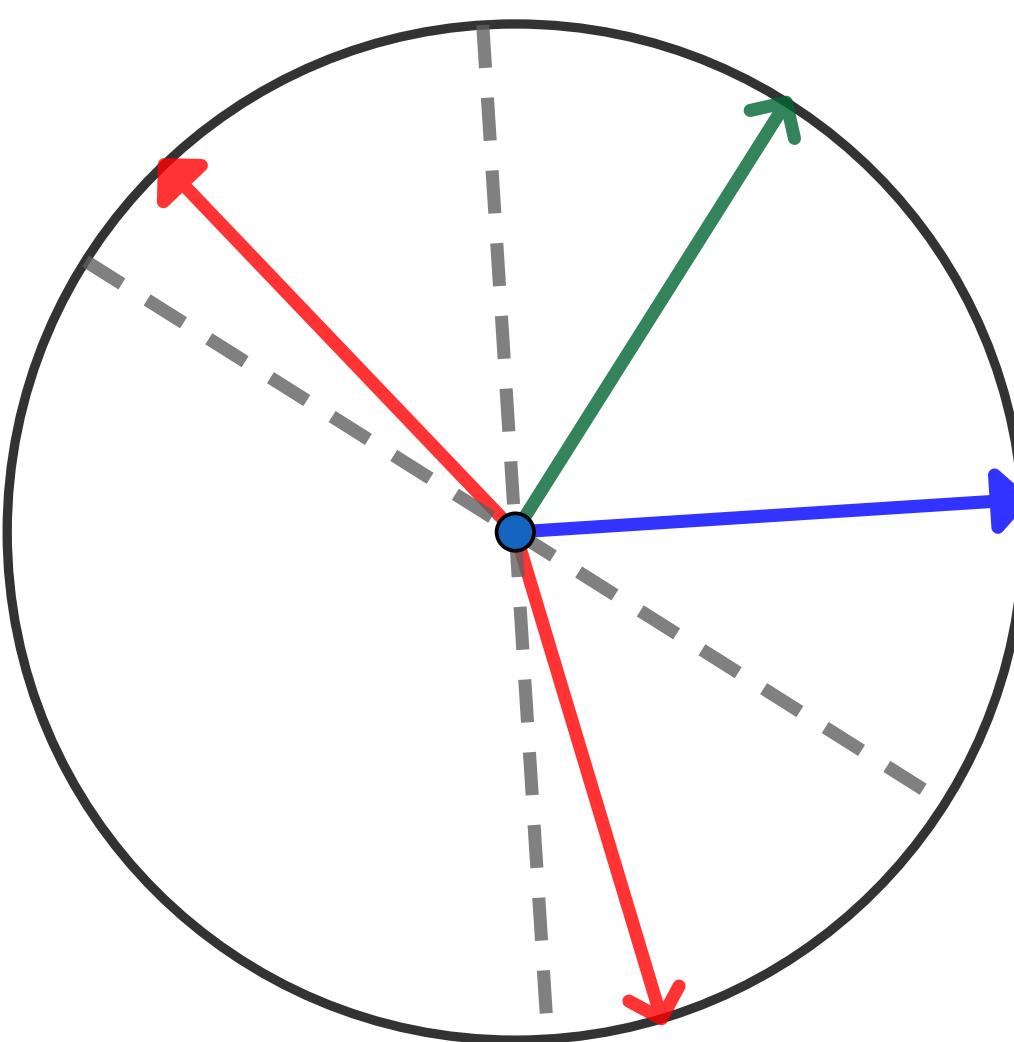
Cosine Similarity的哈希：SimHash

- 类似于MinHash， 我们想找到一个 h , 使得 $\Pr[h(x) = h(y)]$ 可以反映 $\sigma(x, y)$
- 算法：
如何生成：生成 d 个独立的标准正态变量，然后将该随机向量归一化
- 生成一个 d 维随机高斯向量 w (等价于在 d 维单位球面取一个均匀随机点)
 - $h(x) := \text{sgn}(\langle w, x \rangle)$, 其中 $\text{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$ 是符号函数
 - Claim: $\forall x, y \in \mathbb{R}^d, \quad \Pr[h(x) \neq h(y)] = \frac{\theta(x, y)}{\pi}$
 $\theta(x, y) \in [0, \pi]$ 是 x 和 y 的向量夹角

验证SimHash主要性质

$$\text{Claim: } \forall x, y \in \mathbb{R}^d, \quad \Pr[h(x) \neq h(y)] = \frac{\theta(x, y)}{\pi}$$

异号：一个夹锐角另一个
钝角形成 2θ 区域，总共 2π



对于d维：

- 异号 $\{w : \langle x, w \rangle \geq 0, \langle y, w \rangle < 0\}$
- 对应于两个超平面相交，二面角 θ
- 总共的二面角取值范围 2π

多次试验取平均值：到Hamming空间的映射

- 类似MinHash，可以取T个独立SimHash取平均值
- 针对 $x, y \in \mathbb{R}^d$ 的估计量： $\frac{1}{T} \sum_{t=1}^T \mathbb{I}(h^{(t)}(x) \neq h^{(t)}(y))$
- 事实上，可以看作是将 \mathbb{R}^d 上的点对应到T维的Hamming！

- $f(x) := (h^{(1)}(x), \dots, h^{(T)}(x))$

T维Hamming上的点是T维的binary string，距离是异或，即 $\text{dist}_{\oplus}(x, y) = |\{i : x_i \neq y_i\}|$

- 上述估计量就等于 $\text{dist}_{\oplus}(f(x), f(y))/T$
- 即： $\text{dist}_{\oplus}(f(x), f(y))/T \approx \theta(x, y)/\pi$

Locality Sensitive Hashing (LSH)

一种狭义定义

sim(x, y)就是 x 与 y 的相似度度量

- 称hash h 是一种LSH, 若 $\Pr[h(x) = h(y)] = \text{sim}(x, y)$
 - 我们希望相似的元素故意追求哈希冲突, 即放到同样的bucket里
 - 对于不相似的元素避免哈希冲突, 要放到不同的bucket里
- 验证: MinHash和SimHash都是LSH
 - MinHash h : $\Pr[h(S) = h(T)] = J(S, T)$
 - SimHash h : $\Pr[h(x) = h(y)] = 1 - \theta(x, y)/\pi$

一般LSH到Hamming空间的映射

实际中可以用各种非严格随机的哈希，或Universal Hashing

- 考虑一个映射到 $\{0,1\}$ 的随机哈希 $b : X \rightarrow \{0,1\}$
- 若 $p \neq q$ 则 $\Pr[b(p) = b(q)] = 0.5$, 否则 $\Pr[b(p) = b(q)] = 1$
- 考虑将 b 与 h 复合 $b(h(\cdot))$, 则 $\Pr[b(h(x)) = b(h(y))] = \frac{1 + \Pr[h(x) = h(y)]}{2}$
- 下面可以类似SimHash操作，得到Hamming空间的点
 - 找 T 组独立的 h 和 b , T 维二进制串 $f(x) = (b^{(1)}(h^{(1)}(x)), \dots, b^{(T)}(h^{(T)}(x)))$ 即为Hamming表示

对应于“多次试验取平均值”

对MinHash:

$$2\text{dist}_{\oplus}(f(A), f(B))/T = 1 - J(A, B)$$

Hamming近似最近邻查询： 一个专门算法

目标

- 设有 n 个 d 维的Hamming空间 \mathbb{H}^d 上的点
- 性能要求：
 - 误差参数 $c \geq 1$
 - 预处理时间 $O(dn + n^{1+1/c})$
 - 给定任何 $q \in \mathbb{H}^d$, 可在 $O(n^{1/c})$ 时间找到 c -近似最近邻

即：若最近邻距离是 r , 则返回的点距离 $\leq c \cdot r$

算法思路

即除了 r 个坐标/位置，其余位置都相同；
即“多数”位置都相等

- 对于查询点 q ，设 q 真正的最近邻是 q^* ，并且距离是 r
- 主要观察：对于随机排列的坐标，存在某个 k ：
 - q 和 q^* 的前 k 位完全相等的概率比较大
 - 同时，对任何距离 q 超过 cr 的 q' ，前 k 位完全相等大概率不发生
- 因此：将坐标随机排列后找 q 到数据点的最长公共前缀

也就是说相对于 q^* 来说，
 q' 在“多数”位置上与 q 不相等

这是典型情况，
但高概率则需要采用多个排列！

算法：预处理

- 设数据点集是 $P \subseteq \mathbb{H}^d$, 有 n 个点

实际中经常 N 可以取很小

- 设 $N = O(n^{1/c})$, 找 N 个坐标 d 的随机排列 $\sigma_1, \dots, \sigma_N$

共生成 N 份数据的 copy, 总共 Nn 个点

- 对每个排列 σ_i , 将所有坐标按 σ_i 排列后的 n 个数据点按照字典序排序

↓ ↓
1110
1011
1001
1000
0110
0101
0001

σ_i 排列的是坐标

例如 σ_i 将第 1、3 位对换



↓ ↓
1110
1011
0011
0010
1100
0101
0001

重新排序

1110
1100
1011
0101
0011
0010
0001

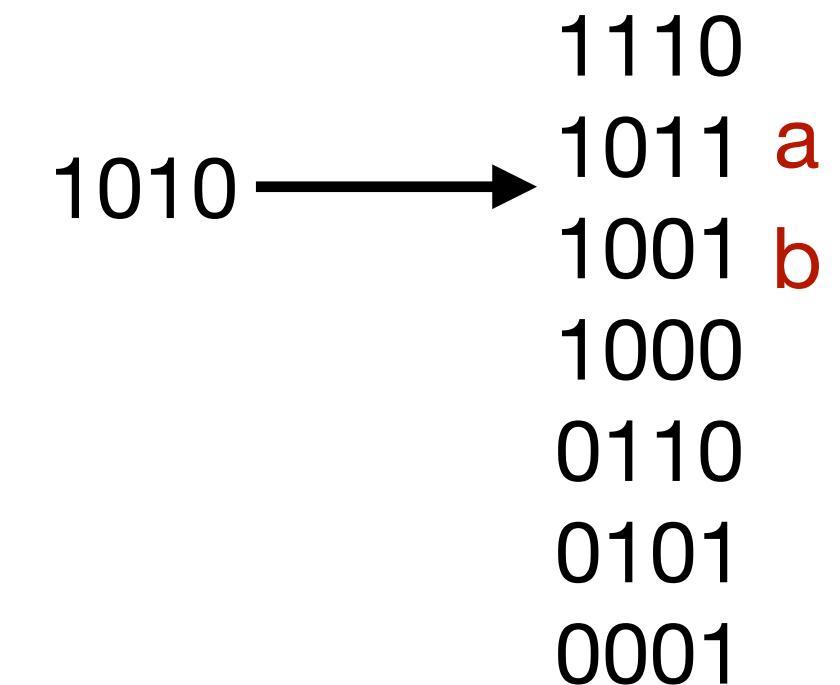
搜索空间大小是 Nn (而不是 n)

算法：给定 $q \in \mathbb{H}^d$, 找 q 的近似最近邻 暴力版本

- 设 $LCP(S, T)$ 为 S 串和 T 串的最长公共前缀
 - 注意: q 和 x 都要用 σ_i 重排所有数位
- 对所有 $1 \leq i \leq N$, 求 $\max_{x \in P} LCP(\sigma_i(q), \sigma_i(x))$
 - 所以 $|S| \leq N$
- 将上述 LCP 中对应的 x 做成集合 S
- 返回 q 与 S 中的 Hamming distance 最近邻作为答案
 - 需要 $|S| \leq O(N)$ 次比较
- 瓶颈: 找 S 需要 $N \cdot n$ 时间, 下页介绍利用预处理的排序降低到 $N \cdot \text{poly}(\log n)$

高效搜索公共前缀最大的Binary String

- 先考虑 $N = 1$, 即只有一个排列。设 n 个 d 维的 binary string 已按字典序排好
- 算法：
 - 二分查找 q , 找到最接近 q 的前后两个 string a 和 b
 - 观察：在有序表中，离 q 越远则与 q 的公共前缀越短
 - 因此可以从 a 向前、从 b 向后每次贪心向公共前缀较大的一个推进
- $N > 1$ 时，每组二分查找 $\sigma_i(q)$ 前后 string a_i, b_i



使用后缀数组等技术
可优化复杂度，但一
般没必要因为 d 比较小

编程实现的一些提示

- 实际应用中长度d比较小，比如 $d \leq 64$ ，因此可以用int64存储

根据之前讨论：对SimHash来说，一般 $O(\log n)$ 足够
- 字典序排序 = 按int64值排序，字典序二分查找 = 按int64值查找
- 求两个int64型a和b的公共前缀长度可以非常高效（下一页）
- 对于典型应用，N可以取很小，比如N = 10或者20

求最高非0位

int32 version

- LCP: $\text{msb}(a \oplus b)$

```
int msb32(unsigned int n)
{
    int res = 0;
    if (n & 0xffff0000) { res += 16; n &= 0xffff0000; }
    if (n & 0xff00ff00) { res += 8; n &= 0xff00ff00; }
    if (n & 0xf0f0f0f0) { res += 4; n &= 0xf0f0f0f0; }
    if (n & 0xcccccccc) { res += 2; n &= 0xcccccccc; }
    if (n & 0xaaaaaaaa) res += 1;
    return res;
}
```

* 参数 $(n^{1/(1+\epsilon)})$ 的选取

细节见Moses Charikar, STOC 2002

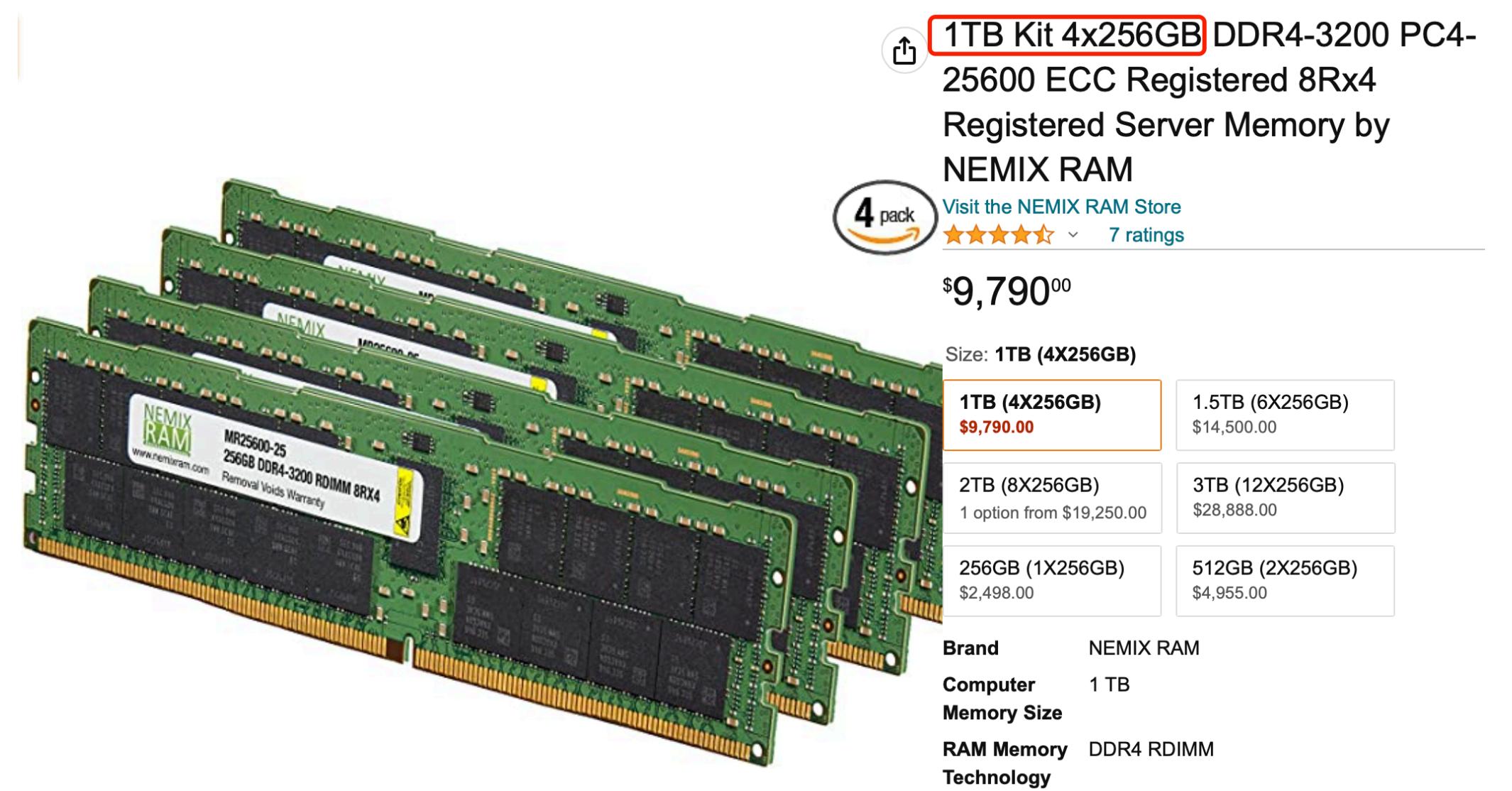
SimHash发明人

- 考虑查询点 $q \in \mathbb{H}^d$, 设 q^* 是 q 的最近点, 并设 $r := \text{dist}_H(q, q^*)$
- 设 $p_1 := 1 - r/d$, $p_2 := 1 - cr/d$, $k := \Theta(\log_{p_2}(1/n))$
- 距 q 大于 cr 的 q' 在每个随机排列中与 q 的前 k 位都相等的概率至多 $p_2^k = 1/\text{poly}(n)$
 - 所以大概率地, 所有排列都没有这样的 q' 会前 k 位相等
- 在某个随机排列中, q^* 与 q 前 k 位都相等的概率是 $p_1^k \approx 1/n^{1/c}$
 - 故 N 个排列中存在一个使得 q^* 与 q 前 k 位相等的概率 $\approx 1 - (1 - 1/n^{1/c})^N = \Omega(1)$

工业界广泛应用

- The algorithm is used by the **Google Crawler** to find near duplicate pages.
- In 2007 Google reported using Minhash and LSH for **Google News** personalization
- In 2021 Google announced its intent to also use the algorithm in their newly created **FLoC (Federated Learning of Cohorts)** system

- Google索引了~50 Billion = $5 \cdot 10^{10}$ 网页，每个网页只需要64bit，共占用 < 500G空间，在单台高端服务器都可以进行全网网页的相似度查询



- 我们介绍的这套xxHash + Hamming最近邻方法获Paris Kanellakis Award

颁给具有重大实际影响力的理论成果

作业九： Hamming空间近似最近邻查询

分值：3分

- <http://cssyb.openjudge.cn/25hw9/>
- Deadline: 4月11日

**推广的LSH定义、多种LSH介绍、
与近似近邻搜索算法**

推广的Locality Sensitive Hashing (LSH)定义

- 我们之前的定义是 $\Pr[h(x) = h(y)] = \text{sim}(x, y)$
 - 不是所有的相似度度量都可以方便地表示成这种形式
 - 更一般地, $\Pr[h(x) = h(y)]$ 仅与 $\text{sim}(x, y)$ 呈现“正相关性”
误差参数 $c \geq 1$
- 推广的LSH解决的是一个 c -近似判定性问题: 给定 r 判定是否存在 cr 以内的近邻
 - 一般的非判定性/优化的近邻查询可以通过二分 r 来解决

一般LSH的定义及基于LSH的近似近邻查询算法

- (c, r, p_1, p_2) -LSH是一族定义在 \mathbb{R}^d 上的随机哈希函数 h , 满足:
 - 对满足 $\|x - y\| \leq r$ 的 x, y , 有 $\Pr[h(x) = h(y)] \geq p_1$ 距离近的点有大概率在同一个bucket
 - 对满足 $\|x - y\| > cr$ 的 x, y , 有 $\Pr[h(x) = h(y)] \leq p_2$ 距离远的点有小概率在同一个bucket

Claim: 定义 $\rho := \frac{\log(p_1)}{\log(p_2)}$, 有 $n^{1+\rho}$ 预处理、 n^ρ 查询时间做 c -近似判定的算法

ρ 成为了关键参数

LSH举例：Hamming空间

Claim: 定义 $\rho := \frac{\log(p_1)}{\log(p_2)}$, 有 $n^{1+\rho}$ 预处理、 n^ρ 查询时间做 c -近似判定的算法

- Hamming空间的LSH
 - 构造: 从 $i \in [d]$ 中均匀随机选取一个 \hat{i} , 然后定义 $h : x \mapsto x_{\hat{i}}$
 - $\Pr[h(x) = h(y)] = 1 - \text{dist}_H(x, y)/d$, 所以 $p_1 = 1 - r/d$, $p_2 = 1 - cr/d$
 - $\rho = \frac{\log(1 - r/d)}{\log(1 - cr/d)} \leq O(1/c)$

(c, r, p_1, p_2) -LSH是一族定义在 \mathbb{R}^d 上的随机哈希函数 h , 满足:

- 对满足 $\|x - y\| \leq r$ 的 x, y , 有 $\Pr[h(x) = h(y)] \geq p_1$
- 对满足 $\|x - y\| > cr$ 的 x, y , 有 $\Pr[h(x) = h(y)] \leq p_2$

LSH举例：单位球上向量的欧氏距离的LSH

- 设数据都是单位球 \mathcal{S}^{d-1} 上的向量，用 ℓ_2 也就是欧氏距离度量距离
- 构造：直接利用cosine similarity的SimHash
- $\Pr[h(x) = h(y)] = 1 - \theta(x, y)/\pi$
 - 所以 $p_1 = 1 - 2 \arcsin(r/2)/\pi, p_2 = 1 - 2 \arcsin(cr/2)/\pi$
 - $\rho = \frac{\log(1 - 2 \arcsin(r/2)/\pi)}{\log(1 - 2 \arcsin(cr/2)/\pi)} \leq O(1/c)$

LSH举例：Cross-polytope LSH

[Practical and Optimal LSH for Angular Distance, Andoni et al.]

- 对单位球 \mathcal{S}^{d-1} 上欧氏距离的情况，有一种SimHash的推广
- 设 $M \in \mathbb{R}^{d \times d}$ 为一个随机高斯矩阵，即每个元素是独立的 $\mathcal{N}(0,1)$
- 对 $x \in \mathcal{S}^{d-1}$ ，哈希函数计算 $y := Mx/\|Mx\|_2$ ，然后找离 y 最近的 $\{\pm e_i\}_{i \in [d]}$ 返回
 - $\{e_i\}$ 是标准单位基
- 当 M 的行数 = 1时退化成SimHash
- 可以得到 $\rho = 1/c^2$ ，比SimHash的 $1/c$ 要好

Cross-polytope的优化实现

- 该哈希中， $M \in \mathbb{R}^{d \times d}$ 为一个随机高斯矩阵，即每个元素是独立的 $\mathcal{N}(0,1)$
- $Mx/\|Mx\|_2$ 可以看作是对 x 进行的“全”随机旋转，但运行时间是 d^2 的
- 此处可以使用“伪”随机旋转矩阵，优化到 $O(d \log d)$
- 伪随机旋转矩阵： HD
 - H 是Hadamard matrix，性质是 Hx 可以在 $O(d \log d)$ 时间计算，并且是正交阵/旋转阵
 - D 是随机的±1对角阵，随机性只有n位，因此是伪随机
 - 优化：用 $HD_1HD_2HD_3x$ 代替 Mx ， D_i 是独立的
 - Andoni et al. 测试过，需要至少3次
 - HD 矩阵在多项相关工作都证明了理论上的有效性

Hadamard矩阵

- $H_1 = [1]$

1/ $\sqrt{2}$ 主要是normalization，对我们来说不重要，可以省去
- $H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}$

$m = \log_2 d$
- 可以使用分治法计算 Hx ，时间复杂度 $O(d \log d)$

可以归纳证明
- 旋转阵要求： $R^T = R^{-1}$ （当然需要normalize一下）， H 是满足该要求的
- 有很多其他应用

实现Claim的算法

假定 p_2, r 是常数 (因此不计)

Claim: 定义 $\rho := \frac{\log(p_1)}{\log(p_2)}$, 有 $n^{1+\rho}$ 预处理、 n^ρ 查询时间做 c -近似判定的算法

- 定义 $k := O(\log n / \log(1/p_2))$, $T := n^\rho$
- 定义 T 个独立哈希 $\{g_i\}_{i \in [T]}$, 其中 $g_i : x \mapsto (h_1^{(i)}(x), \dots, h_k^{(i)}(x))$
- 预处理: 计算 $g_1(P), \dots, g_T(P)$
- 查询 q : 对每个 $i \in [T]$ 找到任意的 $x_i \in P$ s.t. $g_i(x_i) = g_i(q)$, 并返回距 q 最近者

$h_j^{(i)}$ 是一个独立的LSH
 g_i 是 k 个 $h^{(i)}$ 的拼接

g_i 一般是一个int/boolean的vector

正确性分析

以常数概率，下面两个性质成立：

- P1：对所有 q 和满足 $\|x - q\| > cr$ 的 x ，**没有** i 满足 $g_i(x) = g_i(q)$
- P2：对某个 q ，如果有到 q 距离小于 r 的 x ，则**存在** i 满足 $g_i(x) = g_i(q)$

P1和P2推出：若有距 q 在 r 内的点，算法输出一个距离 cr 内的点；否则算法返回空

性质P1

P1：对所有 q 和满足 $\|x - q\| > cr$ 的 x , 没有 i 满足 $g_i(x) = g_i(q)$

- 利用 g 是 k 个 h 的独立拼接
- 考虑 $x \in P$ 满足 $\|x - q\| > cr$
- LSH的保证说明对每个 i 有 $\Pr[g_i(x) = g_i(q)] \leq p_2^k \leq 1/\text{poly}(n)$
- 对所有 x 和 i 用union bound推出：
 - “存在满足 $\|x - q\| > cr$ 的 x 和 $i \in [T]$ 使得 $g_i(x) = g_i(q)$ ”概率为 $1/\text{poly}(n)$

性质P2

- P2: 对某个 q , 如果有到 q 距离小于 r 的 x , 则存在 i 满足 $g_i(x) = g_i(q)$
- 设 x 满足 $\|x - q\| \leq r$
- 对于某个固定的 $i \in [T]$, $\Pr[g_i(x) \neq g_i(q)] \leq 1 - p_1^k = 1 - n^{-O(\rho)}$
- 所以 $\Pr[\exists i, g_i(x) = g_i(q)] = 1 - \Pr[\forall i, g_i(x) \neq g_i(q)] \geq 1 - (1 - n^{-O(\rho)})^T$
- 这个量是常数

如果想达到更高的成功率, 可以略微增大 T , 例如 $T = n^\rho \log n$

作业十：欧氏单位球上的近似近邻查询

分值：3分；截止日期：4月16日

- <http://cssyb.openjudge.cn/25hw10/>
- 可自由尝试：
 - 标程用cross-polytope LSH，用Hadamard优化
 - simhash也可以过
- 可选探究：一般LSH的方法 vs 前面介绍的专用算法来做作业九

关于本题的数据/worst case

- 设 $2^d \gg n$ 。本题数据主要是随机生成的，而这一定程度上也是worst case
 - 即在Hamming cube $\{-1, 1\}^d$ 上随机生成，然后除以 \sqrt{d} 归一化到球面
 - (归一化前) 典型行为：任何两个随机向量的汉明距离都whp在 $\frac{d}{2} \pm O(\sqrt{d})$
 - \sqrt{d} 相对于 $d/2$ 就是“无法区分”，因此是worst-case
 - 但球面上点的最小距离 r 其实很大 (考虑内积：whp两两基本垂直)

* 贪心搜索Heuristic/基于图索引的最近邻查询

预处理时间较长，但换来实际中一般好于LSH的查询性能

预处理：

对于每个数据点 u ，如果 v 是 u 的 k -近邻之一则连有向边 (u, v)

精确构造该图可能需要 n^2 时间，但可以采用若干heuristic来优化

贪心查询过程

设查询点是 q :

随机找一个起点 u

while true

从 u 及 u 的邻居中找距离 q 最近的，设为 v

若找到的 $v \neq u$ 则设置 $u := v$ 并继续，否则break

贪心查询的性能

- 每轮迭代只需要检查 k 个点
- 实际来看while-loop的轮数通常比较小， 算法收敛快
- 找到的近邻多数情况是精确最近邻
- 然而如何对这种行为给出一个好的解释依然没有成熟的研究

一些类比/Intuition

- 社会网络的“六度分隔”现象
- 1960年代Milgram送信实验
 - 随机选择一些“起始人”，要求他们向某个陌生“目标人”转发一封信
 - 知道“目标”的基本信息，比如姓名、住址、职业等，但不能通过邮局/地图送
 - 只能通过自己的熟人经过尽量少的中转把信送到
 - 结果：1/3的信件经过平均6次转发到达了目标！

图的直径比较小，并且支持贪心搜索！

实际网络的一些例子

- <http://snap.stanford.edu/data/index.html>
- <https://www.csauthors.net/distance/paul-erdos/shaofeng-h-c-jiang>

