程序设计实习（实验班-2024春）
课程总结、期末安排

授课教师：姜少峰

助教：冯施源 吴天意

Email: shaofeng.jiang@pku.edu.cn

关于期末考试

时间地点

- 上机考试，闭卷，3小时
- 6月12日上午 8点30 - 11点30
- 地点在理科一号楼1235，与上机位置相同

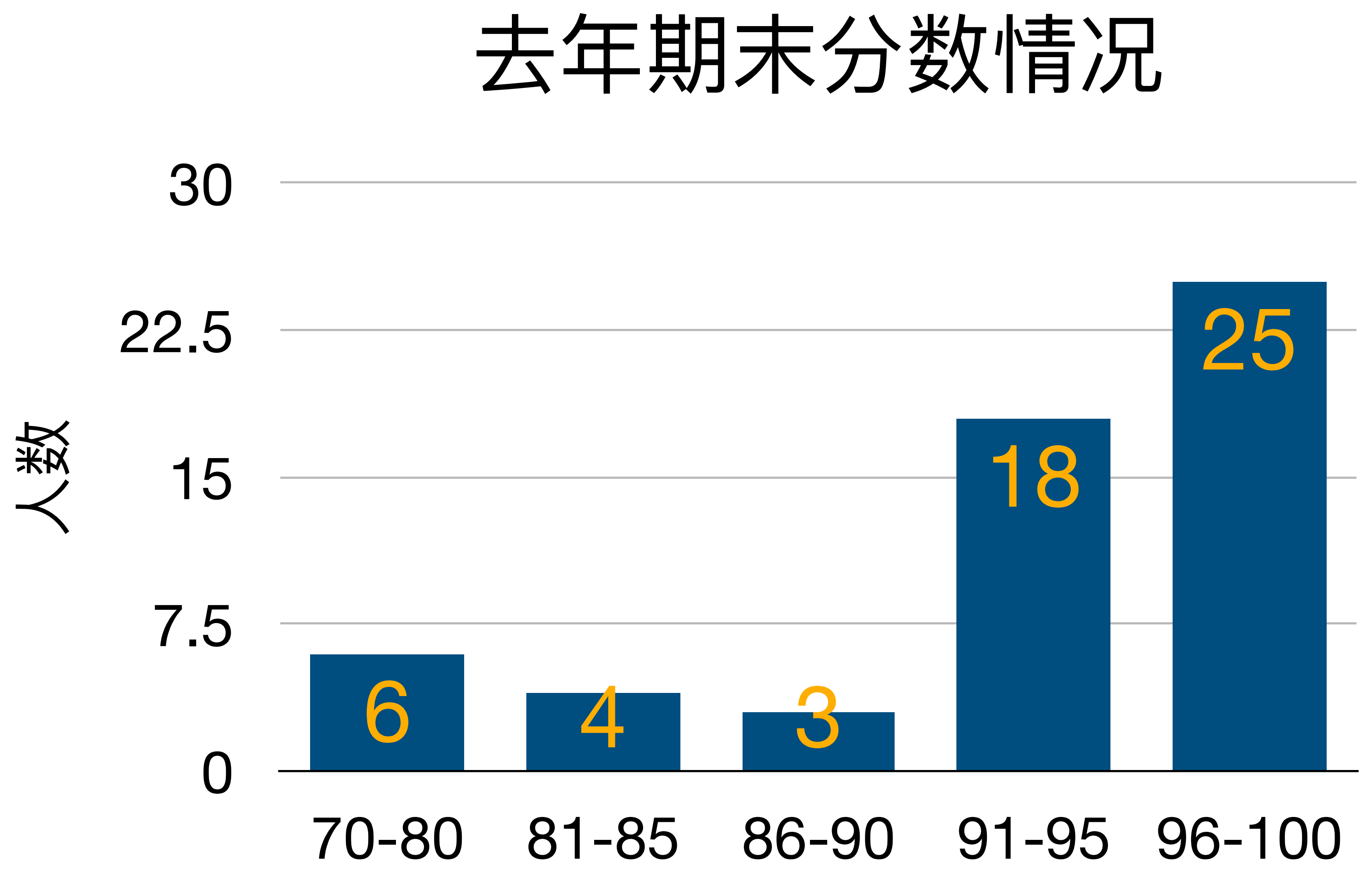
题目构成

- 共7题，按计划总分50（平时分是另50，由19小作业 + 1大作业构成）
  - 1签到题，1作业原题（从Jaccard, 矩阵相乘检查和power method中出一道）
  - 2道中等难度题
  - 3道较难题
- 无C++面向对象编程有关的题目

关于期末评分

- 考前暂不规定特别详细的评分细则，但有一些预期：
  - 若平时满分，则简单题 = 80，+1中等 = 85，+2 中等 = 90，+难题93/96/100

    可认为是一个充分条件
  - 整个评分非线性，过题比较少的同学也不会有特别低的分数
- 最后会根据同学们考后总体表现调整分数
  - 会往有利于同学的方向调
  - 今年在90+分段会更有区分度



课程回顾、主线串讲

大主线

- 程序设计的科学性：现代算法（随机，近似，数据科学，大数据）
  - 随机算法（各种采样），哈希方法：CountMin，MinHash，SimHash
  - 低维数据：几何近似算法
  - 高维数据：降维、利用稀疏性
  - 大数据上的计算：亚线性模型（亚线性时间/查询，数据流等）
- 程序设计的工程性：C++的语言特性，面向对象的设计模式

随机算法

随机算法

- 如何衡量一个随机算法的性能?
  - 精度 vs 失败概率，即$\Pr[\text{ALG is correct}] \geq 1-\delta$
- 随机算法的一般设计方法
  - 设计一个无偏估计，即让算法的数学期望性能达到精度要求
  - 得到常数概率的成功率
  - 多次独立试验提高成功率

Median Trick

- 现在有一个黑盒能以$p > 0.5$概率正确回答某个Yes/No问题的答案
- 如何将该概率强化成任意的$1-\delta$?
- 期望看到：若重复T次，正确答案会占多数，即超过pT个
- 算法：重复T次，选取占多数（或者处于中位）的答案
- T选多大才够?

再论失败概率δ

- 在之前的slides，我们讨论了如何选取δ
- 事实上：常数δ就通常“不失一般性”
  - 经过$O(\log n)$次重复，可以达到$1/\text{poly}(n)$的失败率

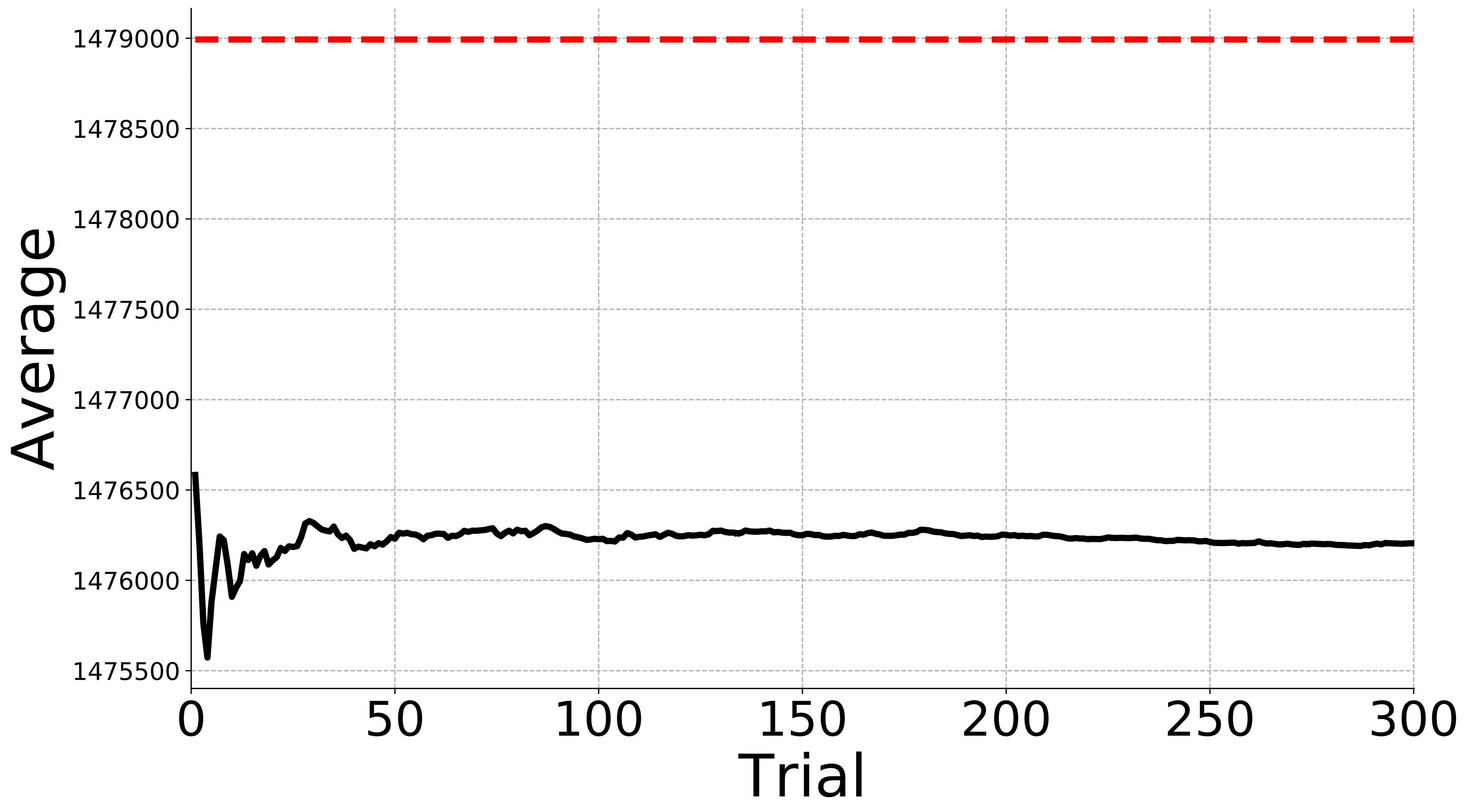
这对于多次运行也通常足够了，毕竟一般只需要运行$\text{poly}(n)$次

- 多大的常数?
  - 如max-cut等可以取最大来放大成功率的：可以是任何常数
  - 如果需要用median trick，则必须是 > 0.5（注意严格不等号）

作业一：最大割随机近似算法实现、调优

分值：2分

- 作业一是一个实验报告（作业要求在教学网和课程网站）
- 给定测试数据，实现课上讲的最大割算法，并绘制重复次数-平均代价图



建议用python的matplotlib画图
作业题目里给出了画图的样例代码

作业二：最大割

分值：1分

- 实现最大割算法，使得在多组数据上测试都必须输出0.45近似的割
- 在openjudge评测，只有所有测试用例都通过才算通过
- 标程使用固定重复次数在设定时间内可以通过
- http://cssyb.openjudge.cn/24hw2/

快速测试矩阵相乘结果是否正确

- 给出三个n x n实矩阵A B C，测试是否AB = C
- 确定性/暴力算法：$O(n^{\omega})$时间，现在$\omega \approx 2.37188$

达到$\omega = 2$是重大open question

- $O(n^2)$时间随机算法：如果AB = C那么一定返回Yes；但是$AB \neq C$时可能答错
  - 随机选取一个n维随机{0, 1}向量x
  - 测试是否ABx = Cx：是则输出YES否则NO

可以在n^2时间计算ABx：先算y = Bx，再算Ay

作业三：测试矩阵相乘是否正确

分值：1分

- 实现课上讲的随机检测方法
- 在openjudge评测，只有所有测试用例都通过才算通过
- 标程使用固定重复次数在设定时间内可以通过
- http://cssyb.openjudge.cn/24hw3/

Rejection Sampling

1/3的解法

- 考虑抛两次的情况，HH TH HT TT四种可能，以不出现TT为条件，则剩下每种出现的概率就是1/3的

TT被reject掉了

- 算法：尝试连续抛掷两次硬币，若TT则重新抛，否则当HH时返回1其他返回0
- 需要抛掷多少次？（需要分析数学期望；最坏情况可以抛任意多次）

本质上：rejection sampling实现了条件概率，但如果reject太多那么会影响性能

比如只有1%的概率不reject，那就大约采样100次才能停下来

作业四：Rejection Sampling

分值2分

- 题目：推广刚刚的rejection sampling，对于一般的一个目标概率p生成两点分布
- 本题为代码填空/交互题，需要使用我们提供的随机性（即{0, 1}均匀分布）
  - 请不要自己在函数中利用其他随机数发生器
- http://cssyb.openjudge.cn/24hw4/

一个基于采样的亚线性算法

只需要$O(1/\epsilon^2)$次采样就能得到$\pm\epsilon n$位误差的估计

- 给定一个很小的误差限ϵ（例如0.01）
- 算法：均匀独立采样$T := O(1/\epsilon^2)$次，找这个采样上的中位数并返回
- Claim：以大常数概率，算法返回的数排在$(0.5 \pm 2\epsilon)n$位上

注意到$0.5n$位是“精确解”，这里有一个$2\epsilon n$位次的误差

作业五：亚线性时间估算分位点

分值：2分

- 作业题推广到一半的p分位点（中位数是$p = 0.5$的特例）
- 本题为代码填空/交互题，需要用我们提供的查询器来访问数据
- 最后的性能指标不（只）看运行时间，主要看查询次数
- 在标程查询次数的一定范围内都可以通过
- http://cssyb.openjudge.cn/24hw5/

非均匀采样构造数据摘要问题

从均匀采样的局限性说起

- 考虑随机采样的时候，均匀采样是第一个该考虑的
  - 算法设计的一般原则：有简单的就不用复杂的；简单的不够再考虑复杂的

奥卡姆剃刀

- 均匀采样的好处：一般确实可以得到无偏估计，即 期望$\mathbb{E}[X]$ 是正确的
- 不足：未必可以做到 常数概率 落在期望附近！

总结：完整算法、结论和参数选取

问题：这个算法得到的S的大小是多少?

- 算法：
  - 先求使$\mathrm{cost}(A, c)$最小化的c（可以找中位数）
  - 定义环P_i并在每个P_i上依照之前进行m次均匀采样得到S_i，并赋予合适权重
  - 将所有S_i求并集得到S返回
- 保证：对任何q，以概率$1-\delta$有

可以证明一共有$O(\log n)$个环
要对所有环用union bound保证同时成功，需要取
$m = O(1/\epsilon^2 \log(\log n/\delta))$

$$\mathrm{cost}(S, q) \in (1 \pm \epsilon) \cdot \mathrm{cost}(A, q)$$

作业六：d维1-median查询

分值：3分

- 作业为维针对d = 3维的输入的近似查询
- http://cssyb.openjudge.cn/24hw6/
- deadline：3月20日

哈希方法

哈希方法

- Hash = 均匀映射，“最均匀”的是随机哈希
- 直接应用：随机负载均衡，互联网动态缓存算法consistent Hashing

Count-min Sketch

- Count-min sketch是一个数据结构，支持对于任何误差参数$0 < \epsilon < 1$：
  - 插入N个$[n]$上的元素后，给定一个$x \in [n]$，估计x共出现了多少次
  - 具体来说：以大概率满足$C \le \hat{C} \le C + \epsilon \cdot N$

大 = $1 - 1/\mathrm{poly}(n)$

“point query”

$\hat{C}$是估计量，C_x是x插入次数

- 总共使用空间$\dfrac{\mathrm{poly}\log n}{\epsilon}$，且单次查询和插入单个元素时间$\dfrac{\mathrm{poly}\log n}{\epsilon}$

Count-min Sketch

大体思路

- 大致算法：
  - 构造一个随机哈希函数$h:[n]\to[m]$

    $m \ll n$是待定参数
  - 将输入元素均匀映射到m个bucket，对每个bucket $j\in[m]$记元素个数$C[j]$
- 如果没冲突，那么最后对$x\in[n]$查询输出$C[h(x)]$就是精确解
- 冲突的影响：多个元素的count累加在一起被报告了出来

  $m \ll n$因此必然有（很多）冲突

  只会造成对结果的高估！

Count-mim Sketch：完整算法

- 初始化：
  - 设置$T = O(\log n)$个独立的随机哈希$h^{(i)} : [n] \to [m]$，这里$m = 2/\epsilon$
  - 初始化T个m元counter $C^{(i)}[1\ldots m] = 0$ （$1 \leq i \leq T$）
- 插/删$x \in [n]$时：

count-min也支持删除

  - 对每个$i = 1,\ldots,T$，将$C^{(i)}[h^{(i)}(x)]$增/减1
- 查询$x \in [n]$时：返回 $\min_{1 \leq i \leq T} C^{(i)}[h^{(i)}(x)]$

应用：Count-min做数据流Heavy Hitter

至多有k个

- k-Heavy hitter (k-HH)：A是长度N元素在[n]上的数组，求出现次数 $\geq N/k$的元素

相关问题：返回top-k出现次数元素
k-HH一定在top-k，但是反之不然！

- 应用
  - A是亚马逊/京东物品成交/访问记录，要快速得到当日最流行爆款商品
  - A是路由器上的IP访问日志，快速找到异常流量防DoS攻击
- 这些应用都是大数据/小内存，因此我们考虑数据流算法
- 数据流：数组A中元素以任意顺序以数据流方式给出，算法在流结束时给出k-HH

一般关注空间复杂度；
处理全流总时间一般追求$n\text{poly}\log n$

数据流可以理解成无随机访问，类似于磁带

数据流近似k-HH：完整算法

- 输入参数：误差$0 < \epsilon < 1$，整数$k \geq 1$，输入元素取值范围n
- 初始化：M = 0，一个在[n]上的count-min数据结构P(ϵ)，一个集合L
- 在数据流中，当$x \in [n]$被插入时：
  - M++；将x插入P；若P.count(x) $\geq M/k$，将x插入L
  - 检查L中所有元素y，如果P.count(y) $< M/k$，则把y从L删除
- 数据流结束时，返回L

要求元素去重

重复元素会被去除

任何时候L中元素count $\geq M/k - \epsilon M$，因此设$\epsilon \leq \dfrac{1}{2k}$，则有$|L| = O(k)$

作业七：k-HH的数据流算法

分值：2分

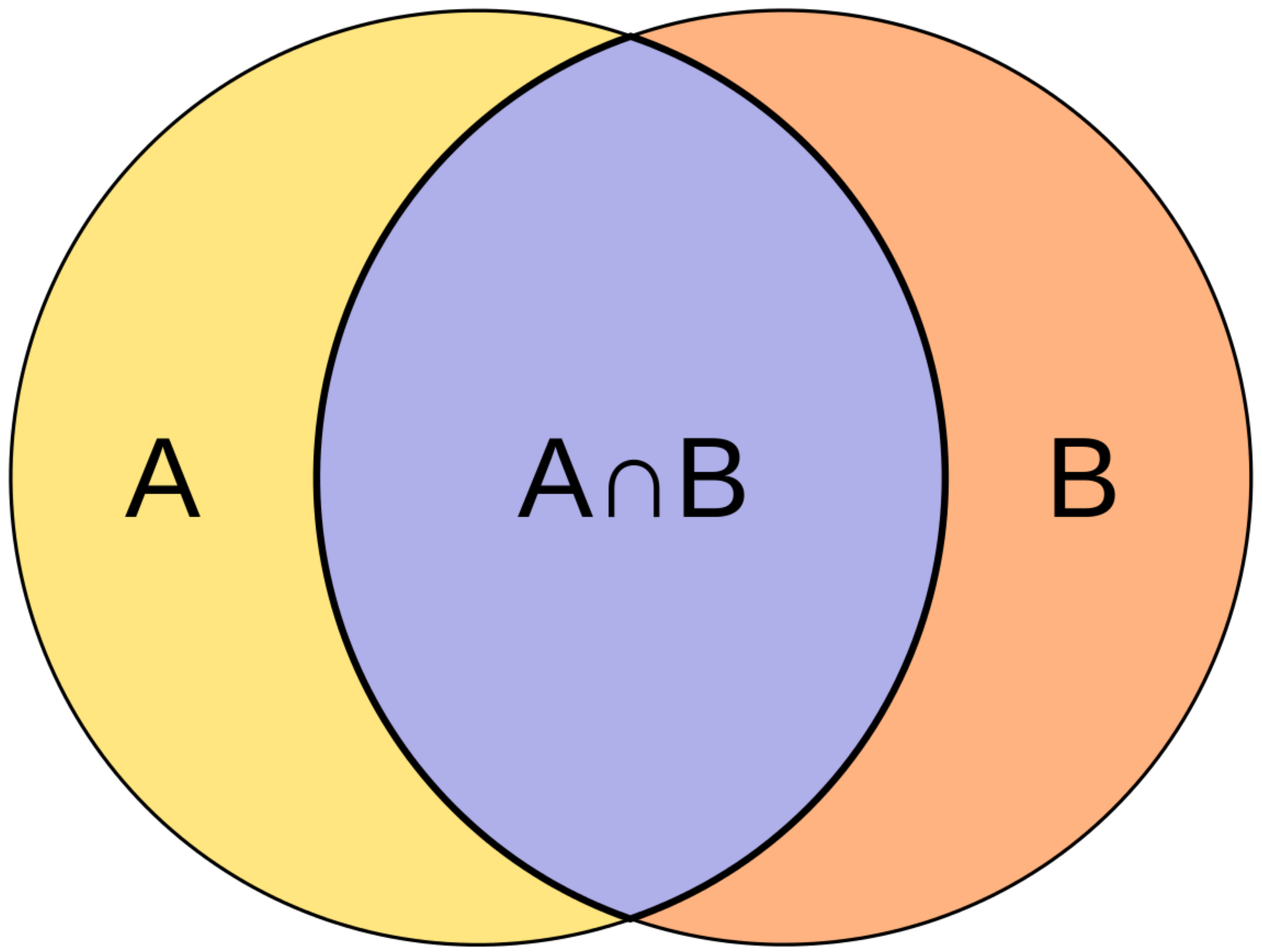
- http://cssyb.openjudge.cn/24hw7/
- 用刚刚介绍的基于count-min sketch的方法解决数据流近似k-HH问题
- 为了采用数据流输入和控制空间，此题需要使用提供的交互库
  - 数据流算法一般分为三个过程：初始化、更新和查询
  - 数据流开始之前运行初始化；数据流插删元素运行更新；数据流结束运行查询
- Deadline：3月27日

Jaccard Similarity

- 两个集合$A, B \subseteq [n]$

是一个[0,1]的数

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



MinHash

- 设所有的集合A_i的元素都来自某个universe [n]
- 设$h:[n] \to [0,1]$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: $\Pr[h_{\min}(A) = h_{\min}(B)] = J(A, B)$

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$h^{(1)}, \ldots, h^{(T)}$
  - 对每个A_i和$h^{(t)}$，计算$h^{(t)}_{\min}(A_i)$
  - 对查询A_i, A_j，计算$\frac{1}{T} \cdot |\{t \in [T] : h^{(t)}_{\min}(A_i) = h^{(t)}_{\min}(A_j)\}|$

如何实现?

$O(|A_i| \cdot T)$时间

计算的是$h^{(t)}_{\min}$相等的t所占比例

$O(|T|)$时间

作业八：用MinHash近似Jaccard Similarity

- http://cssyb.openjudge.cn/24hw8/
- Deadline: 3月29日

Cosine Similarity

- 考虑两个向量$x, y \in \mathbb{R}^d$，定义$\sigma(x, y) := \dfrac{\langle x, y \rangle}{\|x\|_2 \|y\|_2}$

$\sigma(x, y) = \cos(\theta(x, y))$

- 典型应用：文本相似度（TF-IDF）
  - TF = term frequency：一个单词在文档中出现的频率
  - IDF = inverse document freq.：log(总文档数 / 单词出现在多少文档)
  - 对单词w，TF-IDF(w) = TF(w) * IDF(w)，做成一个下标是单词、值TF-IDF向量

一般非常的高维、稀疏

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）
  - $h(x) := \operatorname{sgn}(\langle w, x\rangle)$，其中$\operatorname{sgn}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$ 是符号函数
- Claim: $\forall x, y \in \mathbb{R}^d$，$\Pr[h(x) \neq h(y)] = \dfrac{\theta(x,y)}{\pi}$

  $\theta(x,y) \in [0,\pi]$是x和y的向量夹角

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！
  - $f(x) := (h^{(1)}(x), \ldots, h^{(T)}(x))$
  - 上述估计量就等于$(f(x) \oplus f(y))/T$
  - 即：$\mathrm{dist}_{\mathrm{H}}(f(x), f(y)) \approx \theta(x, y)/\pi$

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

Hamming空间找近似最近邻

- 设有n个d维的Hamming空间$\mathbb{H}^d$上的点
- 性能要求：

  这里典型情况要求$\epsilon \geq 1$

  - 误差参数$\epsilon > 0$
  - 预处理时间$O(dn + n^{1+1/(1+\epsilon)})$
  - 给定任何$q \in \mathbb{H}^d$，可在$O(n^{1/(1+\epsilon)})$时间找到$(1+\epsilon)$-近似最近邻

  即：若最近邻距离是r，则返回的点距离$\leq (1+\epsilon)r$

作业九：Hamming空间近似最近邻查询

- http://cssyb.openjudge.cn/24hw9/
- Deadline: 4月2日

低维：几何近似算法

Idea 1: 格点离散化，线性时间时间$(1+\epsilon)$-近似直径

T可通过选取任意点u，求u到最远点的距离得到（见第一讲）

即满足$1/2 \cdot \mathrm{diam}(P) \le T \le \mathrm{diam}(P)$

- 先O(n)时间找一个直径的2-近似值T

可以是任意确定点

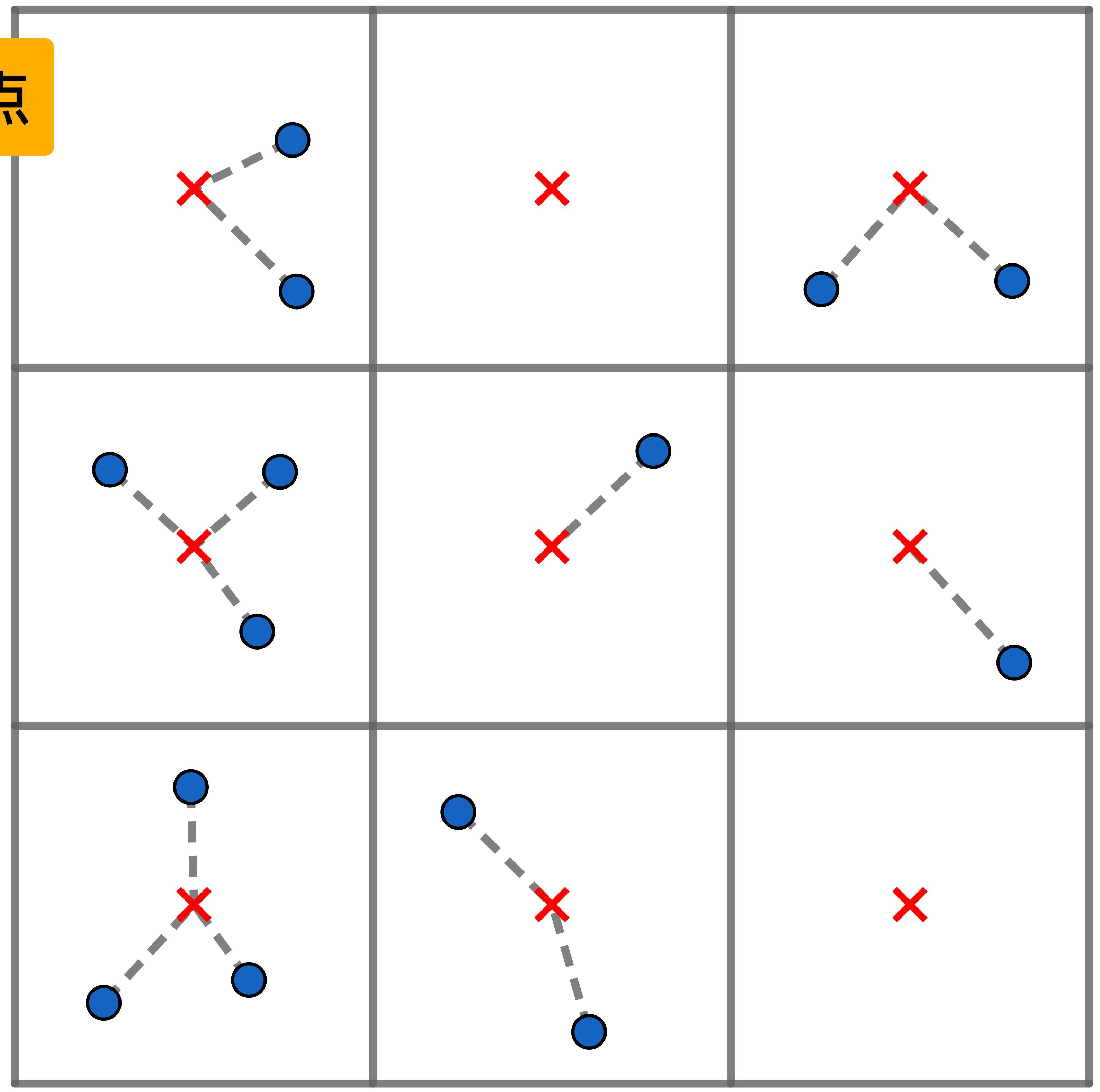
- 作$\ell := \epsilon \cdot T/\sqrt{2}$的网格，并round到中心
  - 因此每个点移动了$\le \sqrt{2}\ell \le \epsilon \cdot \mathrm{diam}(P)$
- 因此新点集P'满足

可在$O(nd\log n)$时间构造P'

$$\mathrm{diam}(P') \in (1 \pm \epsilon) \cdot \mathrm{diam}(P)$$

- 算法：在$|P'|$上暴力求直径，复杂度$|P'|^2 \cdot d$

P'所有点都在$\mathrm{diam}(P') \times \mathrm{diam}(P')$大方格内，小方格$\ell \ge \Omega(\epsilon \cdot \mathrm{diam}(P))$，故$|P'| \le (O(1/\epsilon))^2$

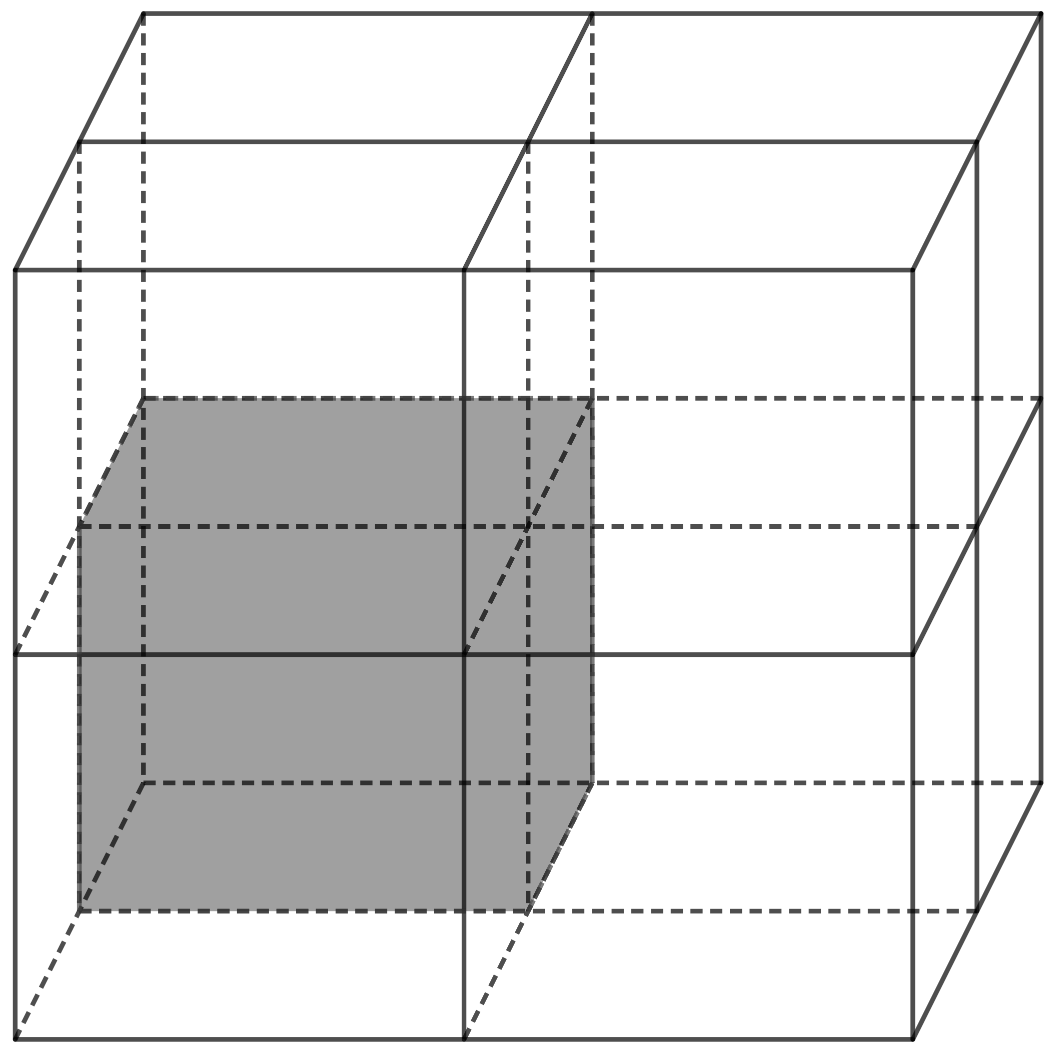


总复杂度：$O(nd\log n) + \epsilon^{-O(d)}$

作业：近似求欧氏点集直径

- http://cssyb.openjudge.cn/24hw10/
- 分值：1分
- Deadline: 4月10日

Idea 2: 递归格点离散化，四分树划分



以二维为例

d维叫“d维四分树”

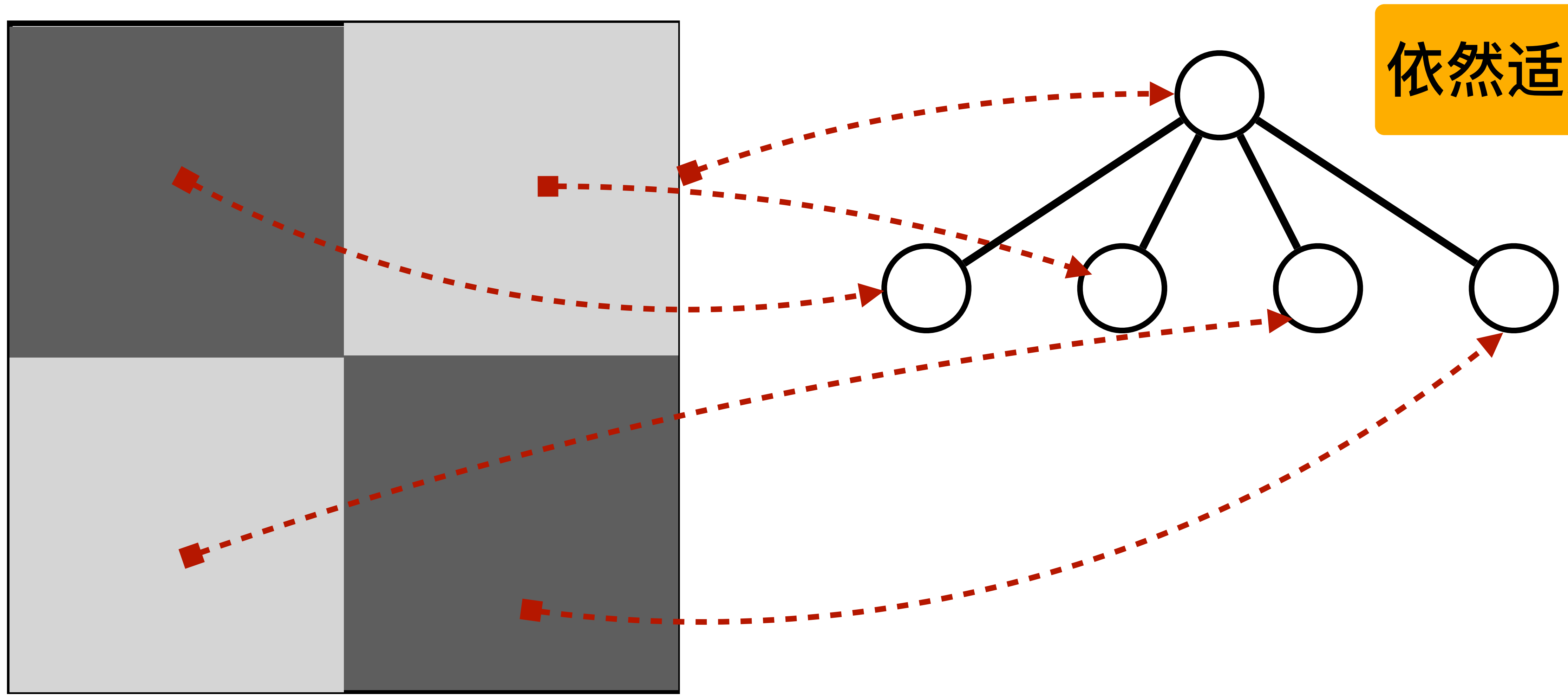
d维是从$[\Delta]^d$划分

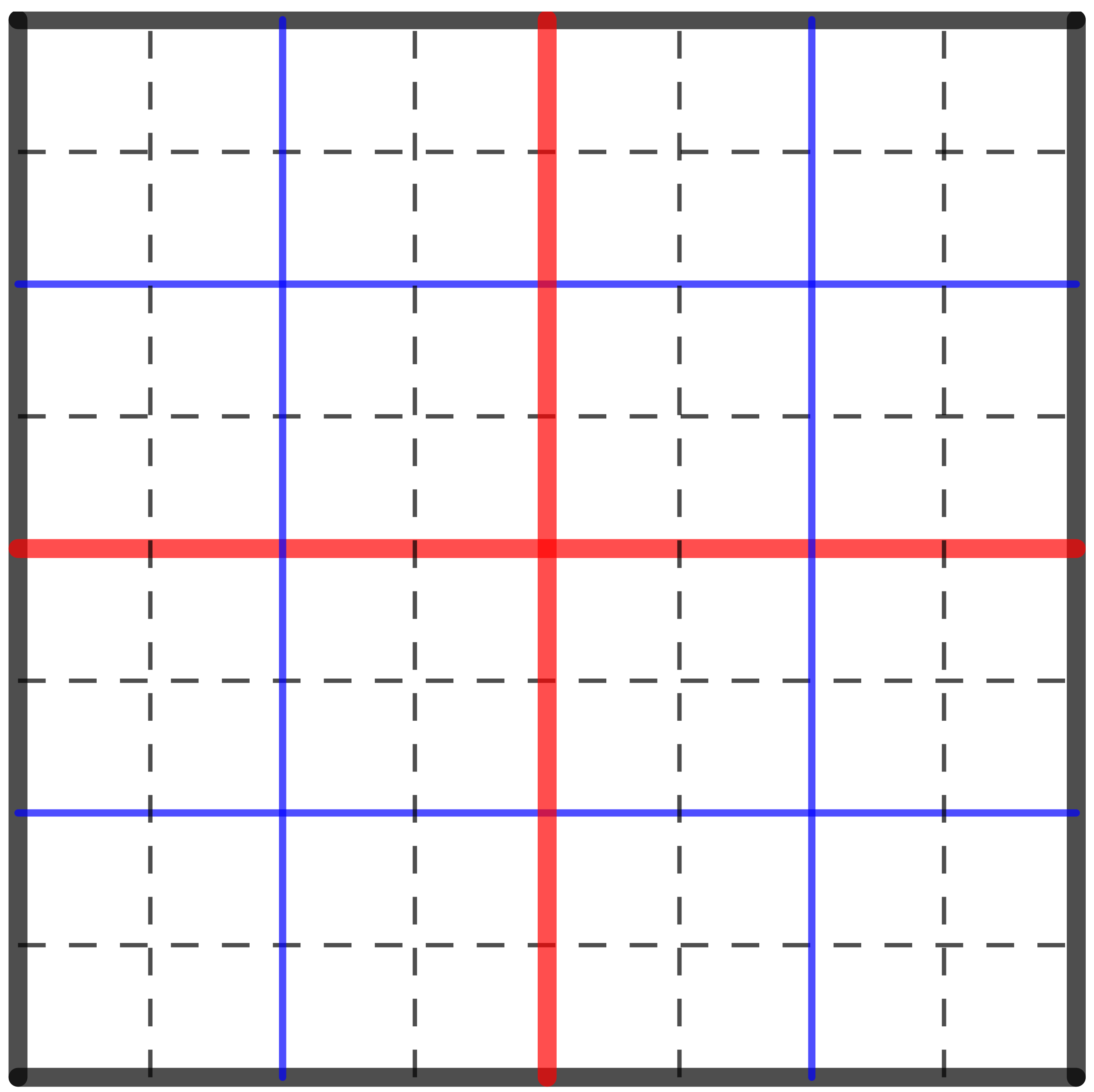
- 从最大的$\Delta \times \Delta$的正方形开始

d维分成2^d个边长为一半的□

- 递归将当前□等分成4个边长为一半的□
- 组织成树形结构，直到只含单点停止，至多$\log_2 \Delta$层

依然适用于d维





伪代码

d维

全局调用 $\square = [\Delta]^d$

BuildTree($\square$, P)

两个停止情况：不含数据点直接返回空；只含1个点返回当前$\square$

if $(P \cap \square = \varnothing)$ return NULL

let T.root = $(\square, \square \cap P)$

除了记录$\square$，也要记录$\square$里含有的数据点集

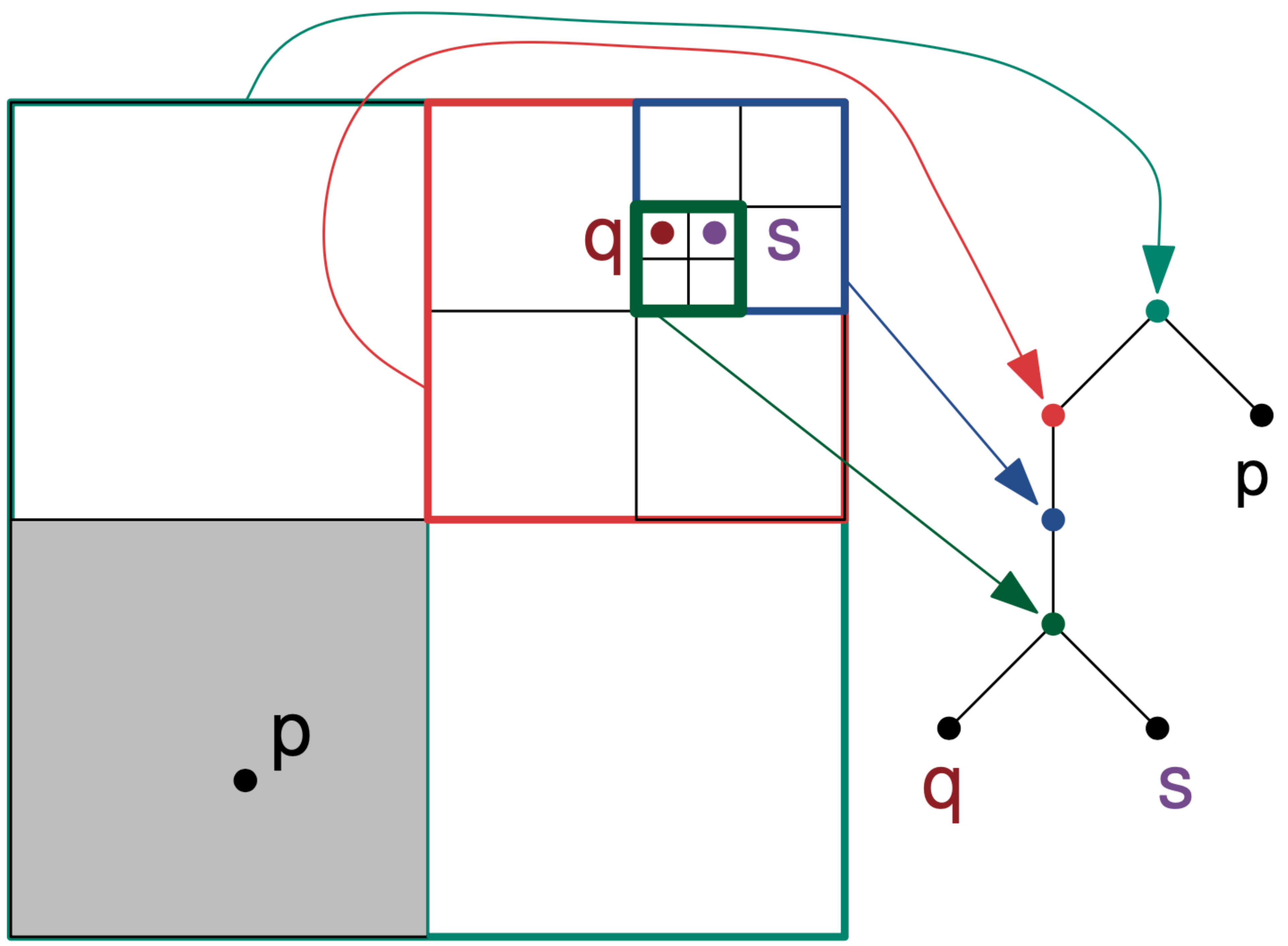
if $(|P \cap \square| = 1)$ return T

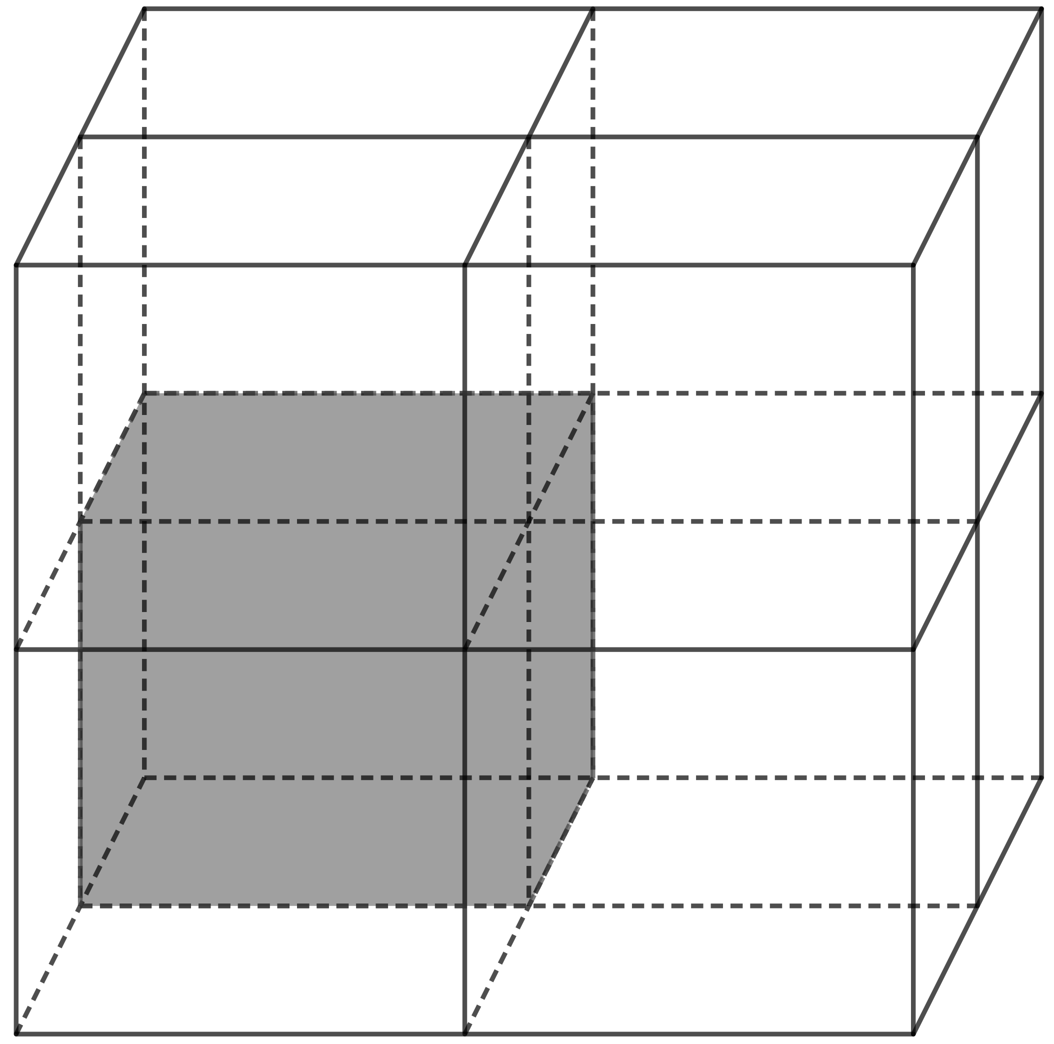
evenly divide $\square$ into 2^d sub-squares $\square_1, \ldots, \square_{2^d}$

for i = 1, …, 2^d, let T.child$_i$ = BuildTree($\square_i, P \cap \square_i$)

return T

整个算法只会产生$O(n \log \Delta)$个非空$\square$（非空即$\square \cap P \neq \varnothing$）





利用四分树进行ϵ-近似最近邻查询

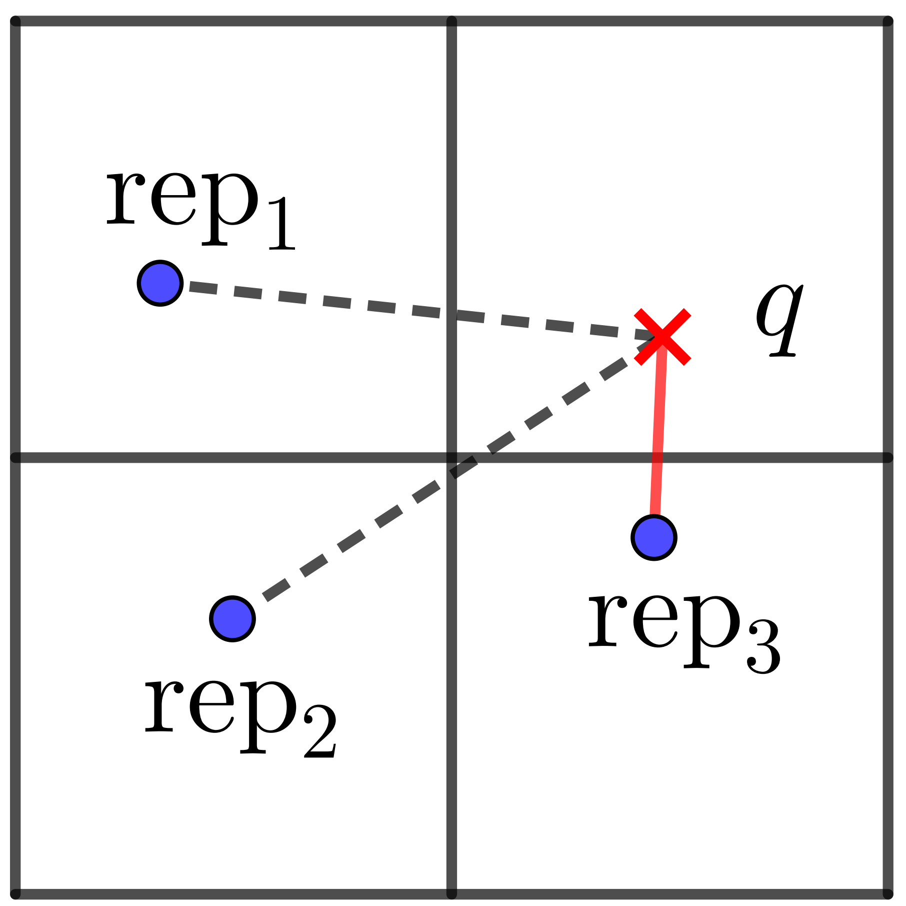
- 问题：
    - 对输入点集$P \subseteq [\Delta]^d$进行预处理
    - 对任意$q \in [\Delta]^d$回答$\hat{q} \in P$，使得$\|\hat{q} - q\| \leq (1+\epsilon) \cdot \min_{q' \in P} \|q' - q\|$

即$\hat{q}$到q的距离是q到q在P最近邻距离的$(1+\epsilon)$倍

- 预处理：BuildTree($\square$, P)，这里调用 $\square = [\Delta]^d$
- 我们将给出一个$O(\epsilon^{-d} \log \Delta)$时间的查询算法返回满足要求的$\hat{q}$

完整算法

输入：查询点q，误差参数$0 < \epsilon < 1$；查询所需时间$O(\epsilon^{-d} \log \Delta)$



设集合A_i为需要检查的四分树第i层的格子的集合（$0 \leq i \leq \log \Delta$）

初始A_0放入最大格子$\square = [\Delta]^d$，$A_i = \varnothing\ (1 \leq i \leq \log \Delta)$，$r = \infty$，$\hat{q} = \text{NULL}$

for $i = 1, \ldots, \log \Delta$

i是层数，按层进行迭代

$\hat{q}$是当前解，r是$\|q - \hat{q}\|$

设S为所有A_{i-1}的格子的2^d个子$\square$的集合

更新当前解为本层最近的rep

对每个$\square_w \in S$，若$\|q - \text{rep}_w\| < r$，更新$r = \|q - \text{rep}_w\|$及$\hat{q} = \text{rep}_w$

对每个$\square_w \in S$，若$r > (1 + \epsilon) \cdot (\|q - \text{rep}_w\| - \text{diam}(\square_w))$，将$\square_w$放入$A_i$

此处是“需要放入$\square_w$”的判据，因此是>

返回$\hat{q}$

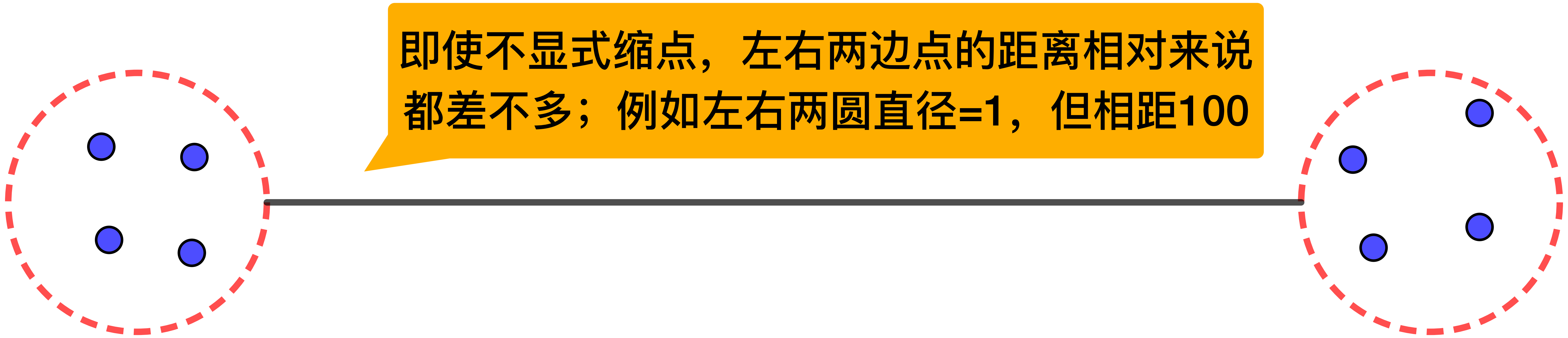
作业十一：欧氏空间近似最近邻查询

- http://cssyb.openjudge.cn/24hw11/
- 分值：2分
- Deadline: 4月10日

Idea 3: 考虑点对的离散化，WSPD

直观讨论

- 输入点集$P \subseteq [\Delta]^d$，设$|P| = n$
- 考虑所有点对的距离（共有$O(n^2)$对），试图将点对距离离散化
- 直觉：设有$S, T \subseteq P$两组点，且他们距离很远
  - 那么可以把S和T分别“缩点”，从而将$S \times T$上的距离等于同一个值



WSPD

定义

- 一个ϵ-WSPD是一系列$\{\{A_i, B_i\}\}_i$，使得

点集之间的距离等于最近点的距离，即
$\mathrm{dist}(S, T) := \min_{x \in S, y \in T} \|x - y\|$

- $\forall i$，$A_i, B_i \subseteq P$且$\max(\mathrm{diam}(A_i), \mathrm{diam}(B_i)) \leq \epsilon \cdot \mathrm{dist}(A_i, B_i)$ 即两组点A_i, B_i距离很远
- $\bigcup_i A_i \times B_i = P \times P$，即任何$p \neq q \in P$可以找到某个$i$使得$p \in A_i, q \in B_i$

可以理解为对$O(n^2)$个距离对的划分

基于四分树的WSPD：完整算法

WSPD用四分树的□表示，总复杂度$O(n\epsilon^{-d}\log\Delta)$，所得WSPD有$O(n\epsilon^{-d}\log\Delta)$项

全局对$\Delta=[\Delta]^d$调用WSPD(□,□)

WSPD($□_u$, $□_v$)

定义：$\delta(□_u)=\mathrm{diam}(□_u)$若$|□_u\cap P|\geq 2$，否则$\delta(□_u)=0$

if ($u=v$ and $\delta(□_u)=0$) return // 单点□不要配对到自己身上

if ($\delta(□_u)<\delta(□_v)$) swap u,v

if ($\delta(□_u)\leq\epsilon\cdot\mathrm{dist}(□_u,□_v)$)

直接看□而不是$P\cap□$

return $\{\{□_u,□_v\}\}$

运行至此，$\delta(□_u)\geq\delta(□_v)$，因此$\{□_u,□_v\}$确实满足WSPD条件，又因为算法会尝试一直递归到底，因此确保最后输出的是合法WSPD

return $\bigcup_{i=1}^{2^d}\mathrm{WSPD}(□_{u_i},v)$

$□_{u_i}$是$□_u$的子□

WSPD应用：精确求最近点对

- 这是一个$O(n \cdot 2^d \log \Delta)$时间的精确算法：
  - 构造$\epsilon = 0.5$的WSPD $\{\{A_i, B_i\}\}_i$

    $O(n \cdot 2^d \log \Delta)$时间
  - 对所有的$\{A_i, B_i\}$，若$|A_i| = |B_i| = 1$，记录A_i和B_i中唯一点的距离

    $O(n \cdot 2^d \log \Delta)$时间
  - 在所有距离中的最近点对就是答案
- 原因：设最近点对是(p, q)且$p \in A_i$，$q \in B_i$，则$|A_i| = |B_i| = 1$

因此算法不会遗漏最近点对

若有多余点x在A_i，则容易看出$\|p - x\| < \|p - q\|$，矛盾

作业十二：最近点对精确求解

- http://cssyb.openjudge.cn/24hw12/
- 分值：2分
- Deadline: 4月10日

WSPD应用：求$(1+\epsilon)$-Spanner

$t \geq 1$

- t-Spanner是数据集$P \subseteq [\Delta]^d$的一个欧氏图的子图H
  - 欧氏图：点集还是P，边集是$P \times P$，边权重是端点间的欧氏距离
  - 子图：点集相同，边集是子集
  - t-Stretch：要求对任何$x, y \in P$有$\|x-y\| \leq \mathrm{dist}_H(x,y) \leq t \cdot \|x-y\|$
  - 追求：给定t，H满足t-stretch条件，且边数尽量少

dist_H代表H上的最短路距离

- 可看作是欧氏图的稀疏“压缩”，后续可使用稀疏图上的算法来解决欧氏问题

Spanner算法

- 算法：先求$\epsilon/8$-WSPD，再对每个A_i，B_i都找代表点rep_1和rep_2连边
- 因此总共是$O(n\epsilon^{-d}\log\Delta)$条边

利用$(1+\epsilon)$-Spanner求$(1+\epsilon)$-近似最小生成树

- 算法：
  - 构造P的$(1+\epsilon)$-spanner H
  - 在H上求MST，则该MST是一个P上的$(1+\epsilon)$-近似的MST
- 解释：
  - 任何P上的MST可以对应于H上一个$(1+\epsilon)$倍权重的子连通图H'
  - 因此H上的MST比H'要小，而H上的MST也是P上的生成树合法解

$\mathrm{MST}(H) \leq (1+\epsilon)\cdot \mathrm{MST}(P)$

H是$P\times P$的子图，因此H中的边都在P的欧氏图中

作业：$(1+\epsilon)$-近似欧氏最小生成树

- http://cssyb.openjudge.cn/24hw13/
- Deadline: 4月17日

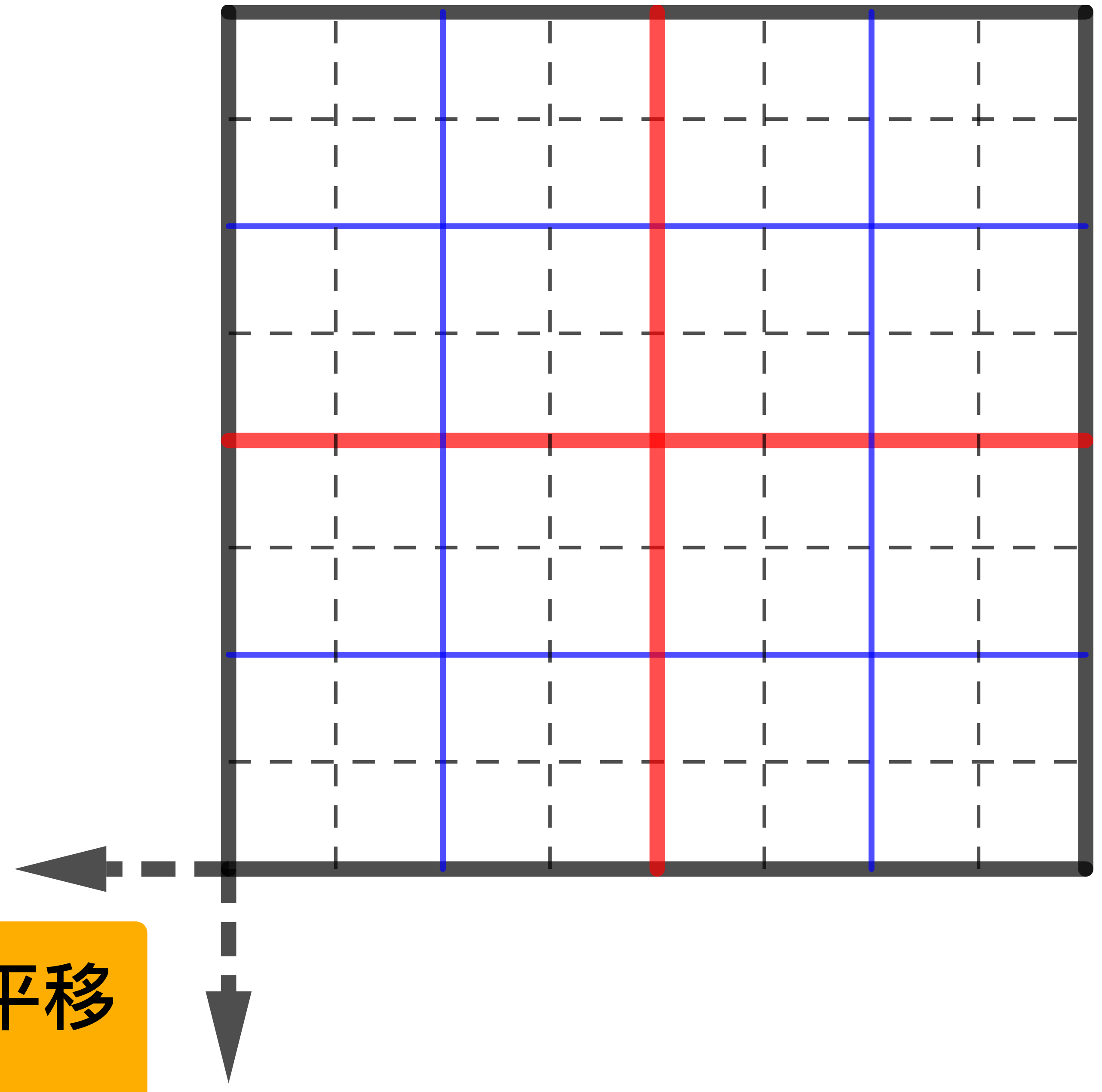
Idea 4: 随机平移四分树与Tree Embedding

- Tree embedding：将数据集P从$\mathbb{R}^d$映射到一棵树$T(V, E)$上，其中$P \subseteq V$
  - 旨在用树上的距离来“近似”/“代替”欧氏距离
  - 首先要求$\forall x, y \in P$，$\mathrm{dist}_T(x, y) \geq \|x - y\|$
- 主要性能衡量叫distortion:

$$\max_{x \neq y \in P} \frac{\mathrm{dist}_T(x, y)}{\|x - y\|}$$

是≥ 1的量，越小越好

构造算法和结论



- 选取$[0,\Delta]^d$上的均匀随机向量v，将P所有点平移v
- 构造四分树BuildTree($\square = [2\Delta]^d, P$)

[2Δ]是为了依然包含随机平移后的P

- 定义树T：每个$\square$定义一个树节点，数据点都放在树的叶子上
  - 定义边：每个$\square$连接到所有的直接孩子，赋予$\sqrt{d}\cdot 2^i$的权值，设$\square$边长2^i
- 对于$x, y \in P$（都是树的叶子），距离$\mathrm{dist}_T(x,y)$就是树上的距离
- 结论：$\forall x, y \in P$，$\mathrm{dist}_T(x,y) \geq \|x-y\|$，$\mathbb{E}[\mathrm{dist}_T(x,y)] \leq O(dh)\cdot\|x-y\|$

distortion是随机的

h是四分树的高度

应用

$O(dh)$近似**MST**

- 结论：$\forall x, y \in P$，$\mathrm{dist}_T(x, y) \geq \|x - y\|$，$\mathbb{E}[\mathrm{dist}_T(x, y)] \leq O(dh) \cdot \|x - y\|$
- 考虑高维MST
    - 先建立T，然后在T上找（所有叶子的）MST，记为F_T，并设精确MST解为F^*

观察到$w(F_T) \leq \sum_{x,y \in F^*} \mathrm{dist}_T(x, y)$，所以

$$\mathbb{E}_T[w(F_T)] \leq \mathbb{E}[\sum_{x,y \in F^*} \mathrm{dist}_T(x, y)] \leq O(dh) \sum_{x,y} \|x - y\| = O(dh) \cdot w(F^*)$$

作业十四：$(1+\epsilon)$-近似欧氏最小生成树

- http://cssyb.openjudge.cn/24hw14/
- Deadline: 4月17日

降维：
应对Curse of Dimensionality

第一类降维：JL Transform

概述

[Johnson-Lindenstrauss, 1984]; 简称JL

JL是一种一般的降维方法，保n个$\mathbb{R}^d$上的数据点集P中任何两点之间的距离

给定误差ϵ，JL是一个映射$f:\mathbb{R}^d \to \mathbb{R}^m$，这里$m = O(\epsilon^{-2}\log n)$

- 使得：$\forall x, y \in P, \|f(x) - f(y)\|_2 \in (1 \pm \epsilon) \cdot \|x - y\|_2$; $O(dm)$时间可算$f(x)$

m决定了降维的维度，称为target dimension，是降维的最重要性能指标

相对误差ϵ，是“最好”的近似比
可直接用于很多与距离有关的问题
以得到改进的近似算法

- 输入的d可大可小，目标维度只与$\log n$和ϵ^{-1}有关

尤其对于例如$d = n$时特别有效

这说明：对于近似算法来说，$O(\log n)$维可以说是“不失一般性”

作业十五：JL实际效果实验报告

- 在实际数据集上测试JL算法的性能
- 按照课上讲的，我们需要测试最坏的相对误差
- 但同时，我们也测试平均误差，可以预计平均误差远小于最坏误差
- https://disk.pku.edu.cn/link/AA6228FD511F2C40FA8DC7D1E2ABF91B0A
- 截止日期2024年5月15日

应用：快速Linear Regression

线性回归定义

一般考虑$n \gg d$，n是数据点个数，d是feature个数

- 输入：$(x_1, y_1), \ldots, (x_n, y_n)$，其中每个$x_i \in \mathbb{R}^d$，$y_i \in \mathbb{R}$

y是一个“label”，即x对应的“结果”

- 要求：找到一个$w \in \mathbb{R}^d$使得$\sum_i (\langle x_i, w \rangle - y_i)^2$即最小二乘误差最小化
  - 解释：想要找到一个y和x之间的线性关系

即希望X与Y线性相关

- 即设x_i, y_i分别是从某个X, Y分布上采样的，那么希望有$Y = w_0 + \sum_{j=1}^{d} w_j \cdot X_j$

这里w_0项是“常数项”，但可以不单独写在求和外面，因为可以给X增加第0维，取值恒等于1，并把w_0可看作是$w_0 X_0$

Linear Regression求解公式

- 将输入$(x_1, y_1), \ldots, (x_n, y_n)$写成矩阵$X \in \mathbb{R}^{n \times d}$第$i$行是$x_i$，向量$y := (y_1, \ldots, y_n)$
- 那么问题可以写作$\arg\min_{w \in \mathbb{R}^d} \|Xw - y\|^2$
- 一般公式：最优的$w^* = (X^\top X)^{-1} X^\top y$

这里$(X^\top X)^{-1}$未必存在，存在条件是X的d列线性独立，即d个feature是独立的

也是通过对每个w_i求导后令导数 = 0得到方程，然后这是方程的解

- 什么复杂度？$X^\top X$需要$O(nd^2)$，是主要复杂度项（另外，求逆需要$O(d^3)$）

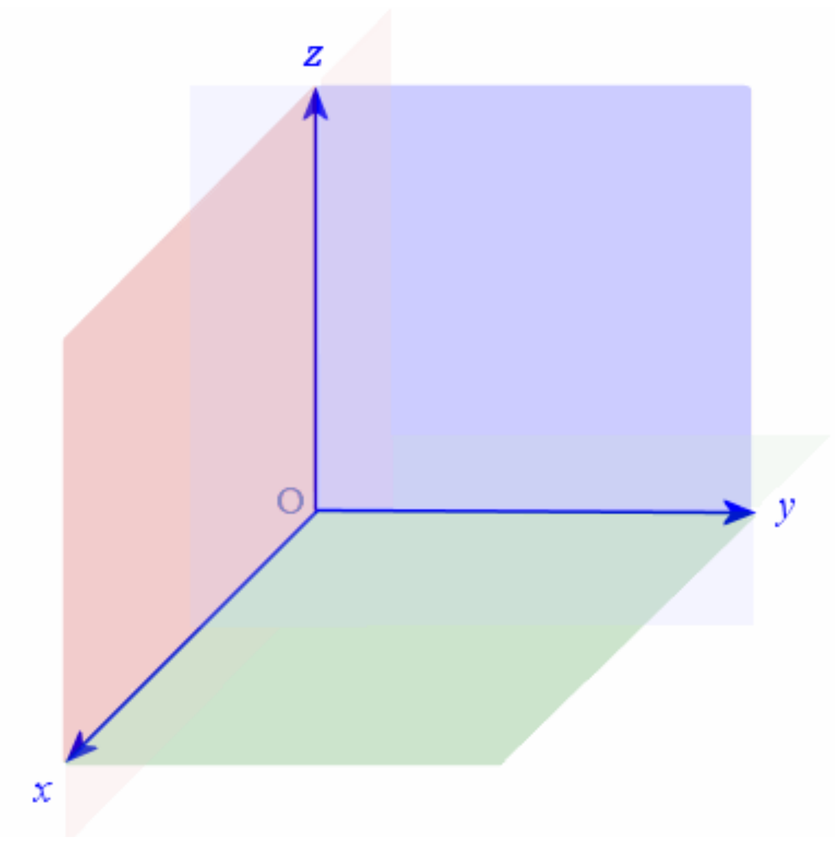
加速方法：Subspace版本的JL

这是因为Xw中w是d维的

- 特殊性：需要考虑的$v = Xw - y$虽然是n维，但rank/独立变量数/自由度只有d
- 设$\mathscr{U}$是$\mathbb{R}^n$的一个d维的子空间

在我们应用中是所有$w \in \mathbb{R}^d$在变换Xw下的集合，即$\{Xw : w \in \mathbb{R}^d\}$
可以（不失一般性）看作是n维空间只有前d个维度非0的子集

例如$\mathbb{R}^3$的x-y平面就是一个2维子空间，并且任意$\mathbb{R}^3$上的2维子空间都可以通过旋转平移得到x-y平面



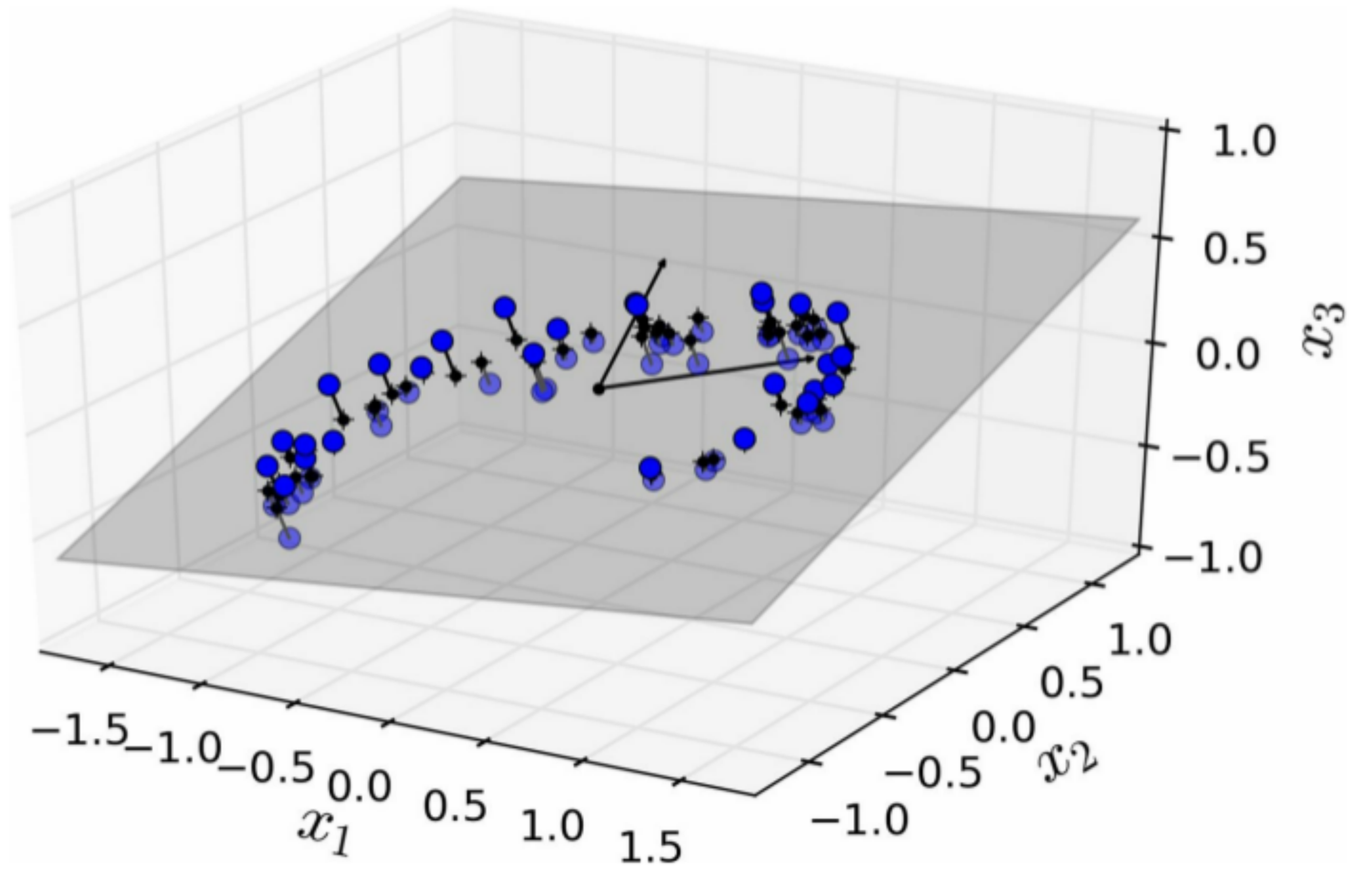


Figure 8-2. A 3D dataset lying close to a 2D subspace

$Xw - y$可等价写作$X'w'$：X'和w'在d维基础上加一维

$$X' = \begin{bmatrix} X_{11}, \ldots, X_{1d} & y_1 \\ \vdots, \ddots, \vdots & \vdots \\ X_{n1}, \ldots, X_{nd} & y_n \end{bmatrix} \quad w' = \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ -1 \end{bmatrix}$$

Subspace版本的JL

- 设$\mathscr{U}$是$\mathbb{R}^n$的一个d维的子空间

回忆：A是$\frac{1}{\sqrt{m}} \cdot (w_1, \ldots, w_m)$，其中每个$w_i$是n维高斯或者归一化的$\{-1,1\}^n$独立向量

定理：$m = O\left(\frac{d \log(\epsilon^{-1}) + \log(1/\delta)}{\epsilon^2}\right)$的JL矩阵$A \in \mathbb{R}^{m \times n}$满足：

将ϵ和δ看作常数，这基本上就是$m = O(d)$

$$\forall v \in \mathscr{U}, \quad \Pr[\|Av\|^2 \in (1 \pm \epsilon) \cdot \|v\|^2] \geq 1 - \delta$$

- 如果把这样的m代入，就可将regression复杂度从$O(nd^2)$降到$O(d^3)$

也就是说：再大的n也可以规约到$n = d$的规模解出来！

高效求解linear regression：完整算法

- 根据刚刚提到的方法生成每列只有一个非0元素的$m \times n$矩阵S
- 预处理/计算$A'X$和$A'y$，这里$A' = S$
- 在$\hat{X} = A'X$和$\hat{y} = A'y$的新输入上利用$w = (\hat{X}^\top \hat{X})^{-1}\hat{X}^\top \hat{y}$公式计算近似最优解$w$
- 输出w

m可以灵活选取，未必按照理论上界
但注意：需要$m \geq d$

需要采用高斯消元

作业十六：利用subspace版本JL高效求解regression

- http://cssyb.openjudge.cn/24hw16/
- 截止日期2024年5月29日

内存/缓存的locality影响效率

- 实现时注意矩阵相乘的循环顺序

```
for (i = 1; i <= d; i++)
    for (j = 1; j <= d; j++)
        for (k = 1; k <= m; k++)
            xtx[i][j] += x[k][i] * x[k][j];
```

K

- 一般的a[i][j] += b[i][k] * c[k][j]的循环顺序应该是i k j

第二类降维：PCA/SVD，目标函数

假设均值0且每列归一

输入$\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$，求orthonormal的$m$个向量$\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^d$，最大化

$$\frac{1}{n}\sum_{i=1}^{n} \underbrace{\sum_{j=1}^{m} \langle \mathbf{x}_i, \mathbf{v}_j \rangle^2}_{\text{squared proj. length}}$$

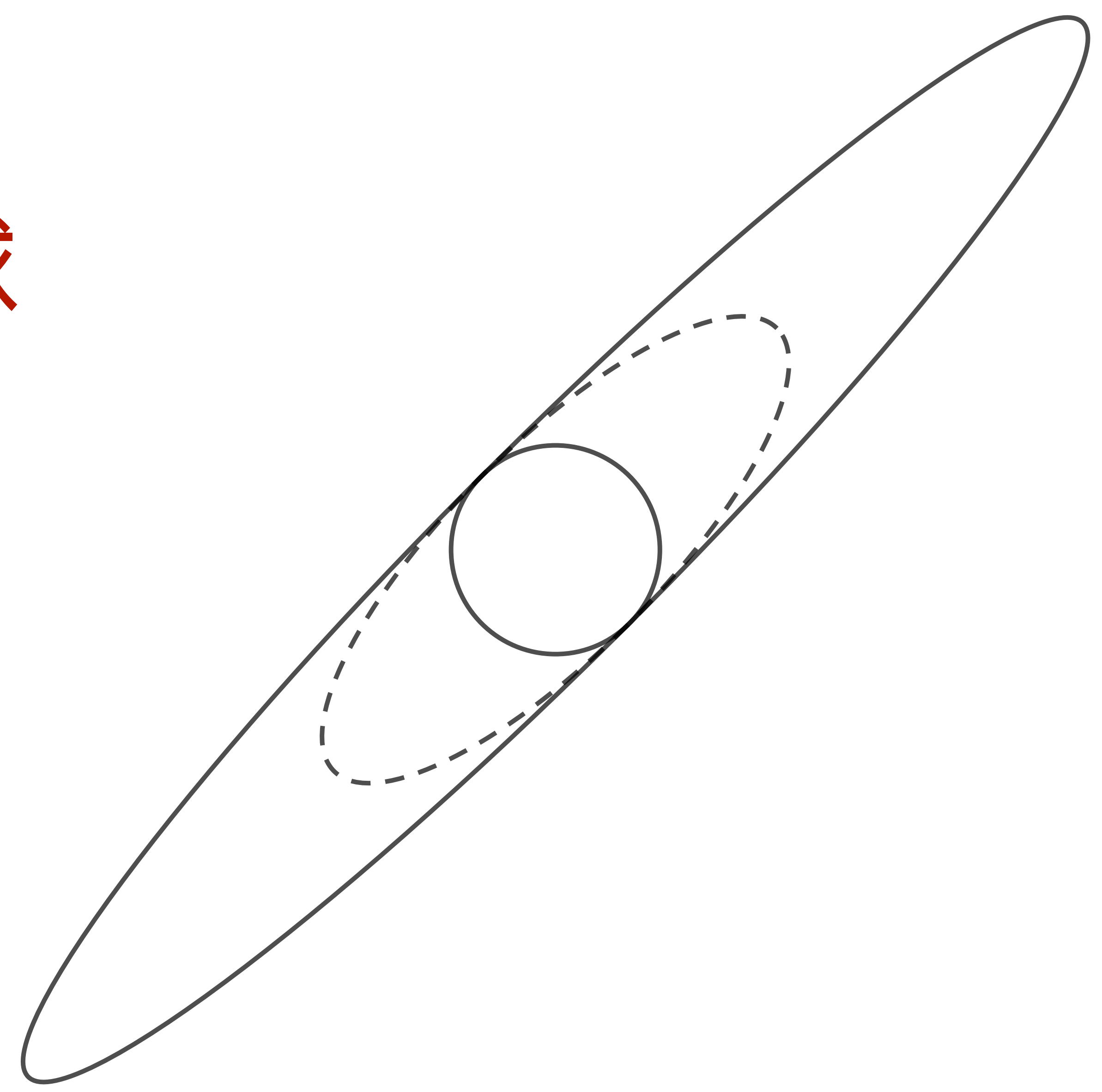
最优的这m个向量叫做$\mathbf{x}_1, \ldots, \mathbf{x}_n$的前$m$个principle components

近似求Top Principle Component: Power Iteration

算法设计思路

- 回忆：$\mathbf{A} = \mathbf{X}^\top \mathbf{X} = \mathbf{QDQ}^\top$本质上是将单位球映射到椭球
  - 椭球的最长轴就对应top principle component
- 任取一个向量$\mathbf{u}$，只要不与$\mathbf{Qe}_1$垂直，那么反复应用$\mathbf{A}$后：
  - 将在$\mathbf{Qe}_1$方向反复被拉伸
  - 若假设最长轴确实比其他轴长，则最后$\mathbf{u}$会非常贴近于$\mathbf{Qe}_1$的方向

$\mathbf{Qe}_1$



如果看椭球，则会在$\mathbf{Qe}_1$上拉的非常长，总体非常扁

算法

输入： $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$

可以每一维独立$\mathcal{N}(0,1)$后归一化生成

选取一个随机方向向量$\mathbf{u}_0$

For $i = 1,2,\ldots$

$\mathbf{u}_i := \mathbf{A}^i\mathbf{u}_0$

根据实际情况设定“≈”的标准

如果$\mathbf{u}_i/\|\mathbf{u}_i\| \approx \mathbf{u}_{i-1}/\|\mathbf{u}_{\mathbf{i-1}}\|$则停止并返回$\mathbf{u}_i/\|\mathbf{u}_i\|$

作业十七：利用Power Iteration求Top PC

- http://cssyb.openjudge.cn/24hw17/
- 截止日期：6月5日

稀疏性

压缩感知具体设定

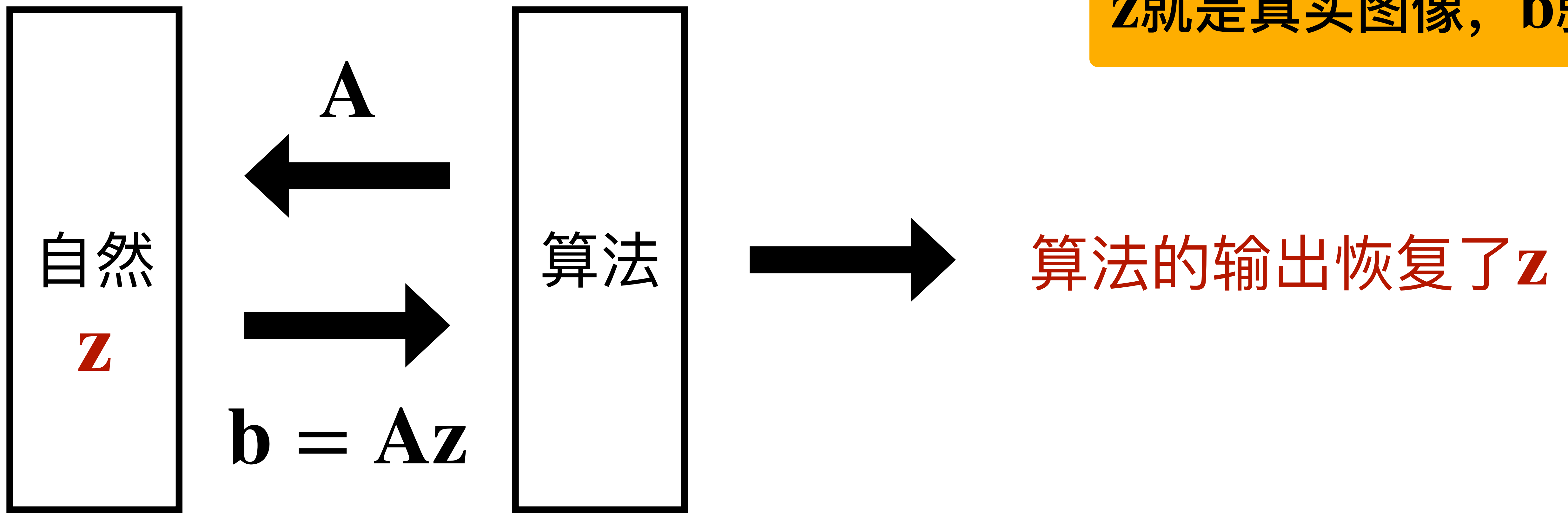
- 我们设计m个linear measurements（线性测量）$\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{R}^n$
- “自然”生成一个未知的信号$\mathbf{z} \in \mathbb{R}^n$

目标是要同一组测量$\mathbf{a}_1, \ldots, \mathbf{a}_m$，对“所有”可能的$\mathbf{z}$都要有效恢复

- 某种硬件告诉我们信号在线性测量上的值$\mathbf{b}$，$b_1 = \langle \mathbf{a}_1, \mathbf{z} \rangle, \ldots, b_m = \langle \mathbf{a}_m, \mathbf{z} \rangle$

想象设计一个影像传感器（摄像头）
$\mathbf{z}$就是真实图像，$\mathbf{b}$就是传感器给出的测量值

- 仅从$\mathbf{b}$恢复$\mathbf{z}$



压缩感知定理：精确版

定理：设$\mathbf{A} \in \mathbb{R}^{m\times n}$其中$m = O(k\log(n/k))$且$\mathbf{A}$的每个元素都从$\mathcal{N}(0,1)$独立采样。则$\mathbf{A}$以高概率，同时对所有的$n$维的$k$-稀疏的$\mathbf{z}$，下面问题的唯一最优解恰好等于$\mathbf{z}$：

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

一般情况：如何处理绝对值

添加辅助变量$\mathbf{y}$来刻画“绝对值”，改写成

$$
\begin{aligned}
\min \quad & \sum_i y_i \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\
& y_i - x_i \geq 0, \quad \forall i \\
& y_i + x_i \geq 0, \quad \forall i
\end{aligned}
$$

否则若$y_i > |x_i|$则可以降低y_i到$y_i = |x_i|$，
不影响其他任何变量，但降低了目标值

在这个新LP中：对每个i，有$y_i \geq |x_i|$，并且由于目标是min，必然有$y_i = |x_i|$

基于压缩感知的Sparse Recovery：完整算法

- 预先给定的参数：k，n，代表信号是n维的k-稀疏的实向量
- 算法先生成一个$m \times n$的随机高斯矩阵$\mathbf{A}$，其中$m = O(k\log(n/k))$
- 算法得到$\mathbf{b} \in \mathbb{R}^m$，使得$\mathbf{b} = \mathbf{A}\mathbf{z}$
- 算法继续计算右边LP的最优解$\mathbf{x}$作为输出

每一维都是独立标准正态$\mathcal{N}(0,1)$

压缩感知定理告诉我们大概率有$\mathbf{x} = \mathbf{z}$

$$
\begin{aligned}
\min \quad & \sum_i y_i \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\
& y_i - x_i \geq 0, \quad \forall i \\
& y_i + x_i \geq 0, \quad \forall i
\end{aligned}
$$

作业十八：压缩感知实验报告

- 本次作业为实验报告，具体要求请见下面链接中的PDF文档：

https://disk.pku.edu.cn/link/AA9B3CB98E746C4490B83C117BA1426408

作业需要调用一个解LP的算法

本次实验报告不限制编程语言和LP solver

  - 可以使用lp_solve（或其他solver），也可以自己实现一个解LP的算法

- 截止日期：2024年6月12日，请提交到教学网！

Sparse Recovery与数据流

重要工具

技术上说，这里允许负频数，并且只要不是0频数，都算作support里面

- 复习：称一个频数向量$\mathbf{x}$是k-sparse的若$\|\mathbf{x}\|_0 \leq k$
- Sparse recovery是一个数据流算法，给定一个参数k：
  - 检测数据流（的频数向量）是否是k-sparse的

    回答Yes/No
  - 如果Yes，就把$\mathrm{supp}(\mathbf{x})$完全恢复出来

    即输出一个集合，等于$\mathrm{supp}(\mathbf{x})$，一共至多k个元素

结论：存在一个高概率成功的$O(k \cdot \mathrm{poly}\log n)$空间的sparse recovery算法

更新时间$\mathrm{poly}\log n$
查询时间$O(k \cdot \mathrm{poly}\log n)$

n是domain大小，也就是频数向量的长度/维数

算法：利用Sparse Recovery解决数据流直径问题

以2倍为步长穷举/猜测直径的值

- for $i = 0, \ldots, O(\log(\Delta))$
  - 维护domain在$\Delta \times \Delta$上的k-sparse recovery structure $\mathcal{S}_i$，参数$k = O(1/\epsilon)^d$
  - 当数据流插/删点p时，找到p所在的$\epsilon \cdot 2^i$格点中心p'，将p'从$\mathcal{S}_i$插入/删除
- 数据流结束时，询问每一个$\mathcal{S}_i$是否k-sparse
  - 找到i值最小的返回Yes的$\mathcal{S}_i$，返回恢复的点集并求该点集直径作为结果返回

刚刚看到，猜测 $\geq \mathrm{diam}(P)$时都会返回Yes
因此需要找符合条件的最小i值来找到 $\approx \mathrm{diam}(P)$的猜测
事实上，可能会找到更小的i，但这只会让解更加精确

作业十九：欧氏点集直径的数据流算法

- http://cssyb.openjudge.cn/24hw19/
- 分值：4分
- 截止日期6月21日

感谢同学们一学期以来的支持！
祝同学们学业进步、前程似锦！