程序设计实习（实验班-2024春）
降维：JL方法

授课教师：姜少峰

助教：冯施源 吴天意

Email: shaofeng.jiang@pku.edu.cn

关于加分

- 参加指定竞赛可以加分
  - 一等奖加3分，二等奖加2分，三等奖加1分
- 指定竞赛目前包括与《程序设计实习》《算法设计与分析》合作的比赛
  - 见比赛宣传
- 其他比赛获奖加分按一事一议情况单独决定

回忆近似求直径的算法

- 我们学过了线性时间近似欧氏点集直径的算法
- 它是否适合于高维呢?
    - 具体来说：运行时间关于维数d的依赖是怎样的?

复习：线性时间$(1+\epsilon)$-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

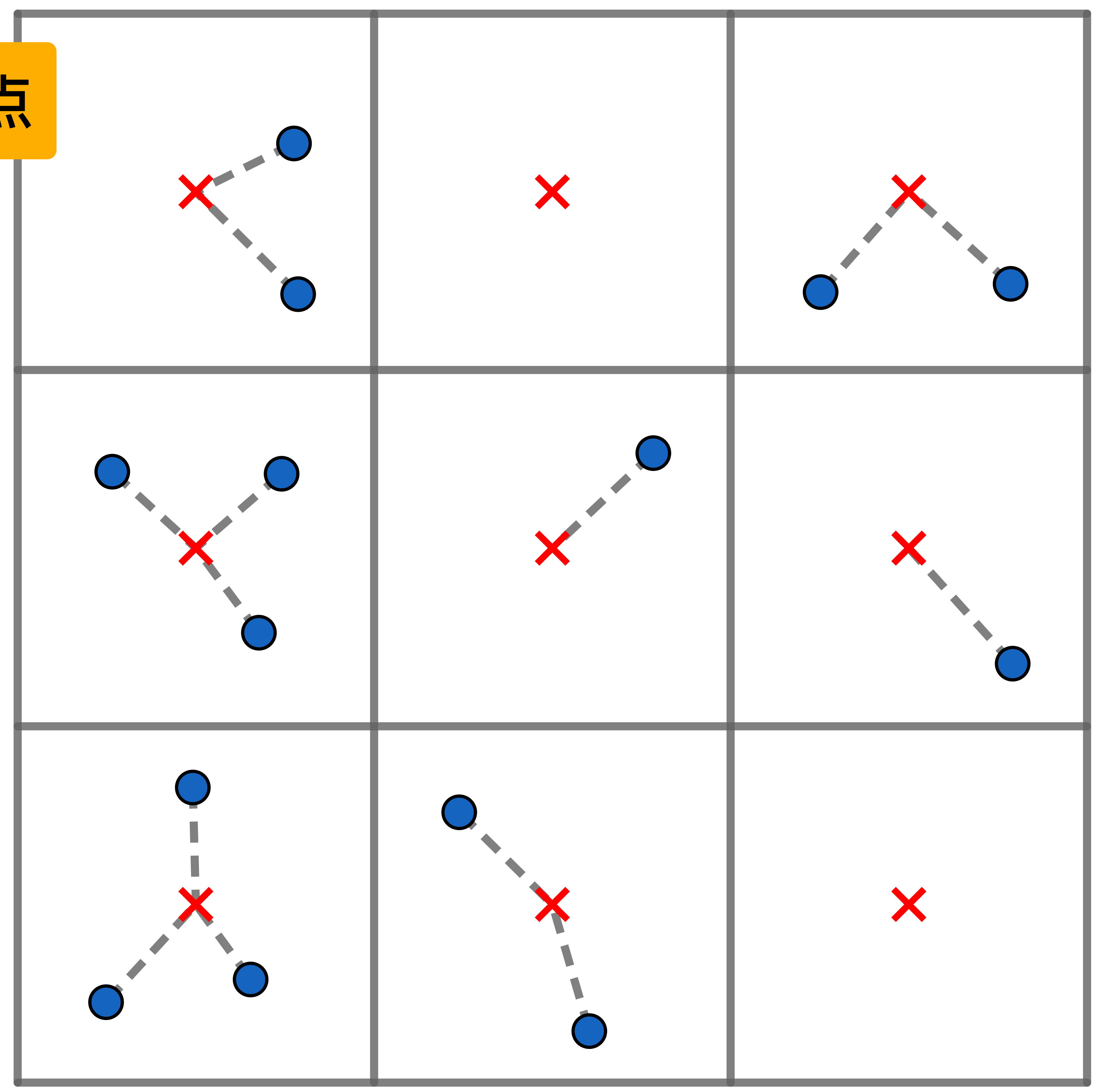
- 作$\ell := \epsilon \cdot T/\sqrt{2}$的网格，并round到中心
  - 因此每个点移动了$\le \sqrt{2}\ell \le \epsilon \cdot \mathrm{diam}(P)$
- 因此新点集P'满足

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：$O(nd \log n) + \epsilon^{-O(d)}$

Curse of Dimensionality

对于计算问题的挑战

- 关于d的依赖经常是指数级的！

需要在某些例如$\mathrm{P} \neq \mathrm{NP}$的复杂度假设下

- 在若干计算模型和问题中，$d = O(\log n)$已经很难！
  - 例如，对于欧氏TSP，无多项式时间$(1+\epsilon)$-近似

[Trevisan, SICOMP 00]

  - 对于最近点对和直径问题，无$n^{2-\delta}$时间$(1+\epsilon)$-近似算法

[Karthic et al., ITCS 19]; [Chen, CCC 18]

  - 对于数据流MST/FL，无$o(n)$空间$(1+\epsilon)$近似

[Chen et al., STOC 23]; [Czumaj et al., FOCS 22]

$d = \Theta(\log n)$已经是高维

很多问题高维下无$(1+\epsilon)$近似

Curse of Dimensionality

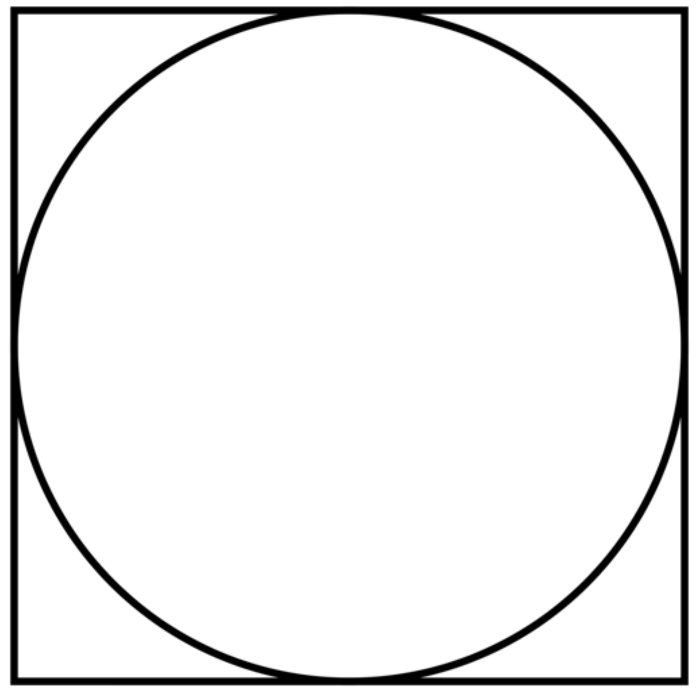
高维欧氏空间的“奇怪”特性

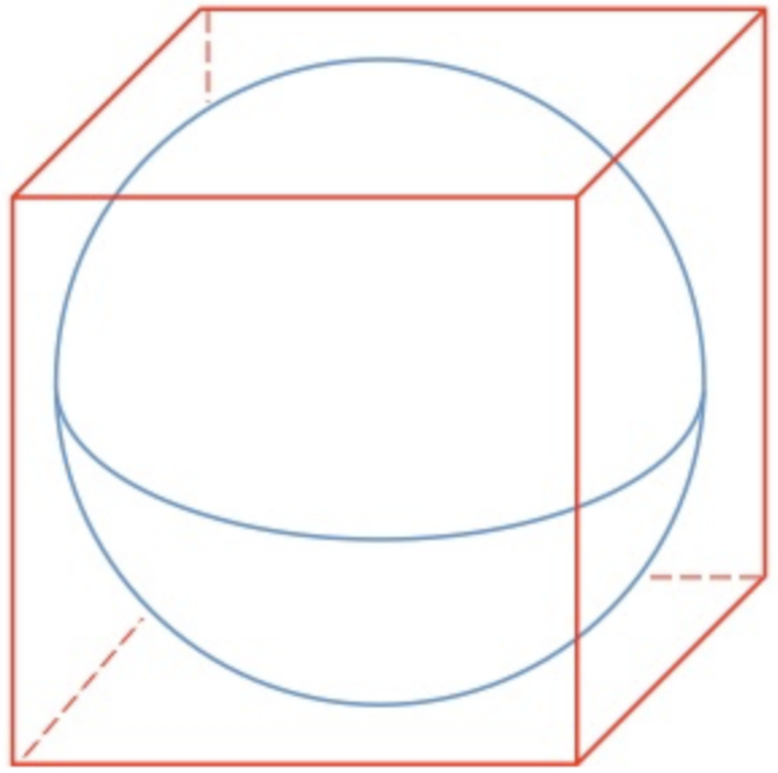
区别于低维空间上的常见直觉

- 高维空间还有很多其他“奇怪”的特性
- 方 vs 圆：单位立方体的体积是1，单位球的体积是$O(2^{-O(d)})$

d维半径为r的球体积近似：$V_d(r) \approx \frac{1}{\sqrt{d\pi}}\left(\frac{2\pi e}{d}\right)^{d/2} \cdot r^d$

- 另：单位圆和方的直径差$\sqrt{d}$倍

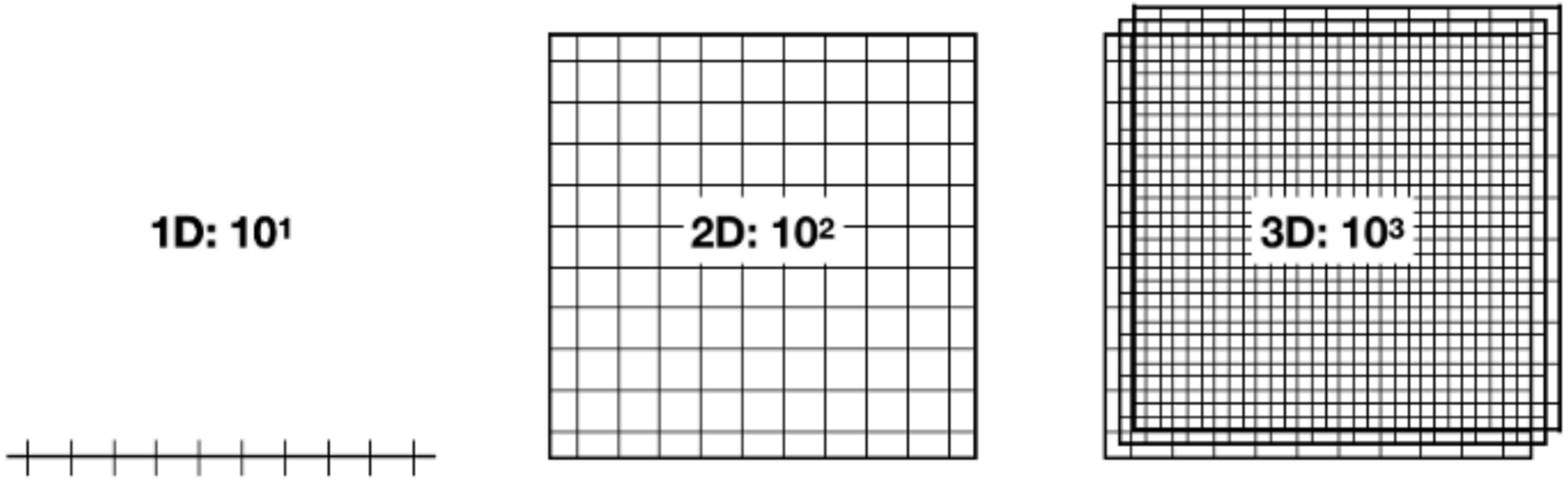




Dimension	Volume of a ball of radius R
0	1
1	$2R$
2	$\pi R^2 \approx 3.142 \times R^2$
3	$\frac{4\pi}{3} R^3 \approx 4.189 \times R^3$
4	$\frac{\pi^2}{2} R^4 \approx 4.935 \times R^4$
5	$\frac{8\pi^2}{15} R^5 \approx 5.264 \times R^5$
6	$\frac{\pi^3}{6} R^6 \approx 5.168 \times R^6$
7	$\frac{16\pi^3}{105} R^7 \approx 4.725 \times R^7$
8	$\frac{\pi^4}{24} R^8 \approx 4.059 \times R^8$
9	$\frac{32\pi^4}{945} R^9 \approx 3.299 \times R^9$
10	$\frac{\pi^5}{120} R^{10} \approx 2.550 \times R^{10}$
11	$\frac{64\pi^5}{10395} R^{11} \approx 1.884 \times R^{11}$
12	$\frac{\pi^6}{720} R^{12} \approx 1.335 \times R^{12}$
13	$\frac{128\pi^6}{135135} R^{13} \approx 0.911 \times R^{13}$
14	$\frac{\pi^7}{5040} R^{14} \approx 0.599 \times R^{14}$
15	$\frac{256\pi^7}{2027025} R^{15} \approx 0.381 \times R^{15}$
n	$V_n(R)$

Curse of Dimensionality
样本稀疏

- 考虑呈每一维坐标的均匀分布以及n个样本点P
  - 有2^d个潜在样本，因此经常$n \ll 2^d$
  - 这导致数据极为“稀疏”，很难形成对整个样本空间的有效覆盖/估计



为什么需要关心?

- 实际数据经常是高维的，例如
  - 文档单词频数向量中，每个维度对应一个不同单词
  - 图像经常展开成每个pixel一维的向量
  - 用户对网站商品的评分向量，每个商品一维

降维

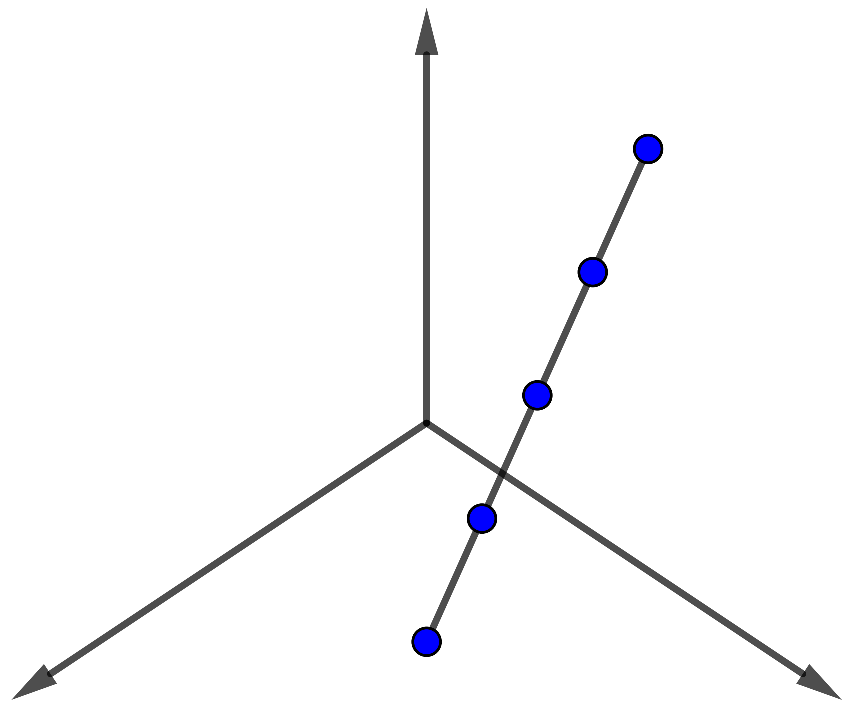
降维概述

- 虽然有curse of dimensionality，但是很多应用中对于高维数据并非无计可施
- 引入近似，利用对距离的近似来降低维度
  - Johnson-Lindenstrauss变换，适用于多种常见数据挖掘/分析问题

我们将讨论如何加速linear regression

- 实际数据并非每维都独立，具有隐含的低维性（只是表现出高维）
  - PCA方法：揭示“主要维度”

可能数据其实是线性的（一维），但可以有很高维的坐标表示
这种高维的坐标表示不应被视为数据真正的维度



针对距离的降维：Johnson-Lindenstrauss

Johnson-Lindenstrauss 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

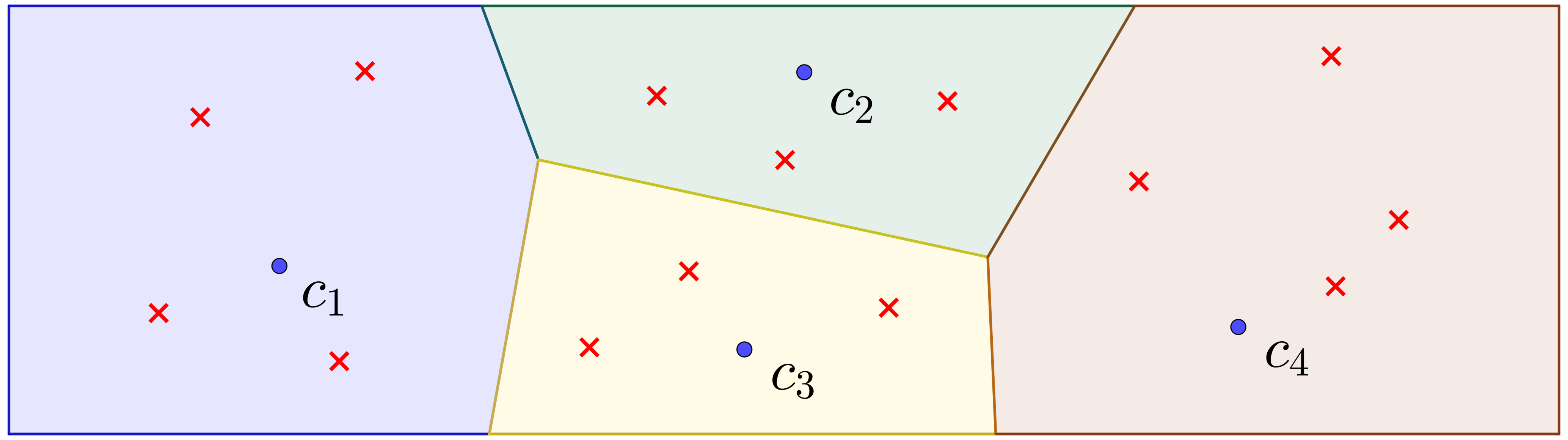
- 输入的d可大可小，目标维度只与$\log n$和ϵ^{-1}有关

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”

JL的直接应用

之后还会介绍一个更加nontrivial的加速线性回归的应用



- 加速算法
  - 有一种聚类问题叫做k-center clustering：
    - 找k个数据点，记为$C \subseteq P$，使到最远数据点的距离最小 $\min_{C} \max_{p \in P} \mathrm{dist}(p, C)$

$\mathrm{dist}(p, C) = \min_{c \in C} \|p - c\|$

  - 著名的Gonzalez算法是2-近似的，运行时间$O(ndk)$
  - 可利用JL（改进）成$O(nd \log n + nk \log n)$

当d较大且$k \gg \log n$是一种显著改进

- 改进存储：如果下游应用主要关心距离，则可以仅存储JL后的向量

Gonzalez算法

对k-center clustering的2-近似

Gonzalez算法[Gonzalez, 1985] 又叫furthest-first traversal

1. 任选一点c_1作为起点，并初始化解$C \leftarrow \{c_1\}$
2. for i = 2 … k
3. 找到C距离最大的点c_i，更新$C \leftarrow C \cup \{c_i\}$
4. 输出C

一共k轮，每轮$O(nd)$，总共$O(nkd)$

为提高运行效率，每次加入c_i时可在$O(nd)$时间预处理每个数据点到C的距离

JL算法

JL构造：总体思路

注意这个2次方：我们本来要的不带2次方

对任意$\epsilon > 0$有$\sqrt{1+\epsilon} \le 1+\epsilon$
和$\sqrt{1-\epsilon} \ge 1-\epsilon$

- 首先注意到：证明$\forall x, y, \|f(x) - f(y)\|^2 \in (1 \pm \epsilon) \cdot \|x-y\|^2$也是够的
- 先针对一对点$x, y \in \mathbb{R}^d$，考虑$v = x - y$
  - 设计一个随机映射$g: \mathbb{R}^d \to \mathbb{R}$使得$\mathbb{E}[g(v)^2] = \|v\|^2$

即g(v)的平方是无偏估计

- 然后多次试验取平均值（但具体操作未必是求和），得到高概率保证
- 最后对于$O(n^2)$对距离使用该结论，使得所有距离保证同时成立

union bound

复习高斯变量/向量的性质

- 高斯向量：每一维都是独立的（一维）高斯$\mathcal{N}(0,1)$
  - 还有一个归一化的版本：归一化后是球面上的均匀随机点，即随机方向
- 两个高斯的和仍是高斯：$\mathcal{N}(\mu_1,\sigma_1^2)+\mathcal{N}(\mu_2,\sigma_2^2)$是一个$\mathcal{N}(\mu_1+\mu_2,\sigma_1^2+\sigma_2^2)$

之前SimHash利用的是该归一化版本

期望和方差的关系对于一般变量也成立，
但高斯变量特殊在：求和依然是高斯！

Random Projection

不同于SimHash：不取符号，而是直接取内积的值

此处与SimHash不同：不要归一化！

- 生成一个$\mathbb{R}^d$上的随机向量w：w每一维是一个独立的$\mathcal{N}(0,1)$
- 对于某个确定的向量$v \in \mathbb{R}^d$，考虑随机函数$g(v) := \langle v, w \rangle$
- $\mathbb{E}[g(v)] = \mathbb{E}[\langle v, w \rangle] = [\sum_{i=1}^{d} v_i w_i] = 0$，没什么用……

$\mathcal{N}(0,1)$变量w_i的性质：$\mathbb{E}[w_i^2] = 1$

- 但是：$\mathbb{E}[g(v)^2] = \mathbb{E}[(\langle v, w \rangle)^2] = \sum_i v_i^2 \mathbb{E}[w_i^2] = \sum_i v_i^2 = \|v\|_2^2$

$(\langle v, w \rangle)^2$是$\|v\|^2$的无偏估计！

多次试验“取平均值”

“拼接”

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

问题：如果不拼接呢？比如 $1/\sqrt{m} \cdot \sum \langle w_i, v \rangle$可行吗？

- 构造m个独立的单位高斯向量$w_1, \ldots, w_m$，定义：

也可以写成$f(v) = Av$，这里 $A \in \mathbb{R}^{m \times d}$是$w_1, \ldots, w_m$构成的矩阵

注意此处是把m个独立的点积**拼接**起来

$$f(v) := \frac{1}{\sqrt{m}}(\langle w_1, v \rangle, \ldots, \langle w_m, v \rangle)$$

即新定义的f是无偏估计

- 计算可得：$\mathbb{E}[\|f(v)\|^2] = \mathbb{E}[g(v)^2] = \|v\|^2$

JL核心结论：

证明推导超出本课范畴，有兴趣可参考其他资料

$$\forall v \in \mathbb{R}^d, \qquad \Pr[\|f(v)\|^2 \in (1 \pm \epsilon) \cdot \|v\|^2] \geq 1 - \exp(-O(\epsilon^2 m))$$

利用union bound

- 对于n个点的集合P，如何应用该结论?

$$\boxed{\forall v \in \mathbb{R}^d, \qquad \Pr[\|f(v)\|^2 \in (1 \pm \epsilon) \cdot \|v\|^2] \geq 1 - \exp(-O(\epsilon^2 m))}$$

- 先考虑（任意）一对点(x, y)，定义$v_{xy} = x - y$并应用结论
  - 此时有：$\Pr[\|f(x) - f(y)\|^2 \in (1 \pm \epsilon) \cdot \|x - y\|^2] \geq 1 - \exp(-O(\epsilon^2 m))$

重要性质：线性变换

$f(x) - f(y) = f(x - y)$

$$\boxed{f(v) := \frac{1}{\sqrt{m}}(\langle w_1, v\rangle, \ldots, \langle w_m, v\rangle)}$$

利用union bound (cont.)

- 我们有了：对于任意确定一对(x, y)：
  $\Pr[\|f(x) - f(y)\|^2 \in (1 \pm \epsilon) \cdot \|x - y\|^2] \geq 1 - \exp(-O(\epsilon^2 m))$
- 我们要的是：每对(x, y)同时保持$\|f(x) - f(y)\|^2 \in (1 \pm \epsilon) \cdot \|x - y\|_2$

Union bound: $\Pr[\bigcup_i B_i] \leq \sum_i \Pr[B_i]$

这个事件的反面就是每对(x, y)同时保持$\|f(x) - f(y)\|$误差在ϵ倍

具体到我们：定义B_{xy}事件为$\|f(x) - f(y)\|^2$误差大于ϵ倍，即“坏”事件

利用union bound：存在一对点发生坏事件B_{xy}的概率不超过$\sum_{x,y} \Pr[B_{xy}]$

- 因此，我们要求$\exp(-O(\epsilon^2 m)) \cdot O(n^2) \leq 0.1$，解得$m = O(\epsilon^{-2} \log n)$

$\Pr[B_{xy}] \leq \exp(-O(\epsilon^2 m))$，且有$O(n^2)$个$B_{xy}$

0.1是一个任意的常数，代表失败概率

重要性质：Data Oblivious

即无论数据集具体的点是什么，都按照同样的方法构造

Data oblivious：指算法/函数的定义不依赖于具体的输入数据集

另一种理解：算法的输入不依赖于整体数据集

回忆：$f(v) := \frac{1}{\sqrt{m}}(\langle w_1, v\rangle, \ldots, \langle w_m, v\rangle)$

- 映射f只用到了随机数w_i，不依赖于数据集

虽然依赖v作为数据点输入，但是并不依赖整个数据集

- 那什么不是data-oblivious的？
  - 例如：求解用m维达到的最优的误差z

$$
\begin{aligned}
\text{min.} \quad & z \\
\text{s.t.} \quad & \|p_x - p_y\|^2 \geq (1-z)\cdot\|x-y\|^2 && \forall x, y \\
& \|p_x - p_y\|^2 \leq (1+z)\cdot\|x-y\|^2 && \forall x, y \\
& p_x \in \mathbb{R}^m && \forall x \in P
\end{aligned}
$$

这里面需要知道数据集P

Data Oblivious的好处

特别适用于大数据/亚线性算法

- 例如对于数据流算法，只需要预先存储m个d维的高斯
    - 每个到来的d维数据点可以在$O(md)$空间直接求出降维后的m维向量
    - 如果不data-oblivious，那么需要读取整个数据集后操作，经常需要线性空间
- 类似地，在并行计算等也可以直接使用

实现细节/其他版本

- 实际应用时很少根据ϵ和n确定维数m，而是直接根据经验指定目标维度
  - 实际表现一般不错；$m = O(\epsilon^{-2}\log n)$只是最坏情况，通常达不到
- 更“简单”的JL：将随机向量w_i替换成每维是独立$\{-1,1\}$均匀随机数的向量
  - 直觉

回忆：$f(v) := \frac{1}{\sqrt{m}}(\langle w_1, v\rangle, \ldots, \langle w_m, v\rangle)$

    - $\{-1,1\}^d$是d维单位立方体，上述随机向量是其随机顶点
    - d维随机高斯是d维单位球上的随机点，是随机方向

用离散的随机单位立方体顶点来近似连续的随机方向

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

平均情况？具体问题？

对一组向量$x_1, \ldots, x_n$，保$\sum_i \|f(x_i)\| \in (1 \pm \epsilon) \cdot \sum_i \|x_i\|$只需要$m = O(\epsilon^{-2})$

对于其他具体问题：

- k-median/k-means：$m = \tilde{O}(\epsilon^{-2} \log k)$ [Makarychev-Makarychev-Razenshteyn, STOC 19]
- Max-Cut：$m = \tilde{O}(\epsilon^{-2})$ [Chen-**J**-Krauthgamer, STOC 23]
- 也有不可能的：保MST和FL到$O(1)$已经需要$m = \Omega(\log n)$

但JL并不能克服本质计算困难

$d = \Theta(\log n)$时：

- 对于欧氏TSP，无多项式时间$(1+\epsilon)$-近似
- 对于最近点对和直径问题，无$n^{2-\delta}$时间$(1+\epsilon)$-近似算法
- 对于数据流MST/FL，无$o(n)$空间$(1+\epsilon)$近似

- 对于上述问题，用JL得到$d = O(\log n)$也没有帮助！
- 进一步：JL的$m = O(\log n)$能改进吗？改进成$m = O(1)$会产生什么矛盾吗？

* JL的确是“紧”的：简单基于体积估计

$m = O(\log n)$是不可避免的

例如三维是$(1,0,0), (0,1,0), (0,0,1)$

原始数据n维：n维标准基，两两垂直并且距离相等（距离是$\sqrt{2}$，当作一单位）

考虑保距离误差$\epsilon = 0.1$倍的降维，设降维后有m维：

- 这些基在保0.1倍距离降维后，可以保证两两距离在$[0.9, 1.1]$单位之间
- 以这些点为球心0.45单位为半径的球互不相交，且包含在直径2单位的m维球内

2 = 0.45 x 2 + 1.1

半径r的d维球体积公式：$f(d) \cdot r^d$

$$n \cdot f(m) \cdot 0.45^m \leq f(m) \cdot 1^m \quad \text{推出} m = \Omega(\log n)$$

所有n个0.45单位的球的体积

直径为2即半径为1的球的体积

应用：linear regression

Linear Regression

线性回归

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

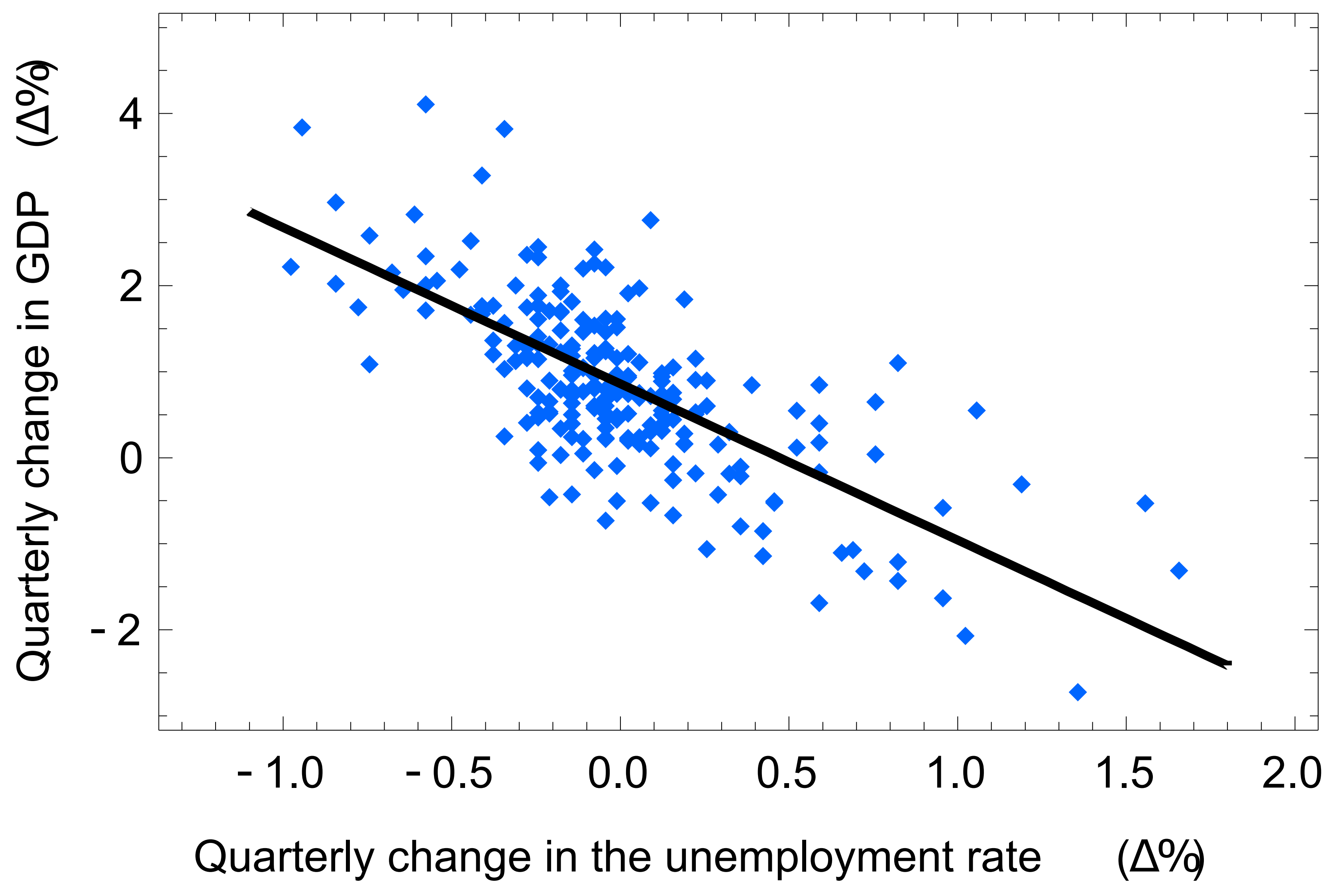
即希望X与Y线性相关

  - 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$w_0 X_0$

举例：$d = 1$

- 考虑$x_i \in \mathbb{R}$的情况，
- 要求$w, w_0 \in \mathbb{R}$最小化$\sum_i (x_i w + w_0 - y_i)^2$
- 如何解？
  - 暴力展开后是关于w和w_0的二次函数
  - 可以对w和w_0分别求（偏）导数并令导数 = 0得到一个方程



$d = 1$情况的求解

$$\sum_i (x_i w + w_0 - y_i)^2 = \sum_i x_i^2 w^2 + w_0^2 + y_i^2 + 2ww_0 x_i - 2x_i y_i w - 2w_0 y_i$$

对w和w_0分别求导并令其 = 0：

$$\begin{cases} 2(\sum_i x_i^2)w + 2(\sum_i w_0 x_i - x_i y_i) = 0 \\ 2nw_0 + 2\sum_i (x_i w - y_i) = 0 \end{cases}$$

一般情况的求解

- 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$\arg\min\limits_{w\in\mathbb{R}^d} \|Xw - y\|^2$
- 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

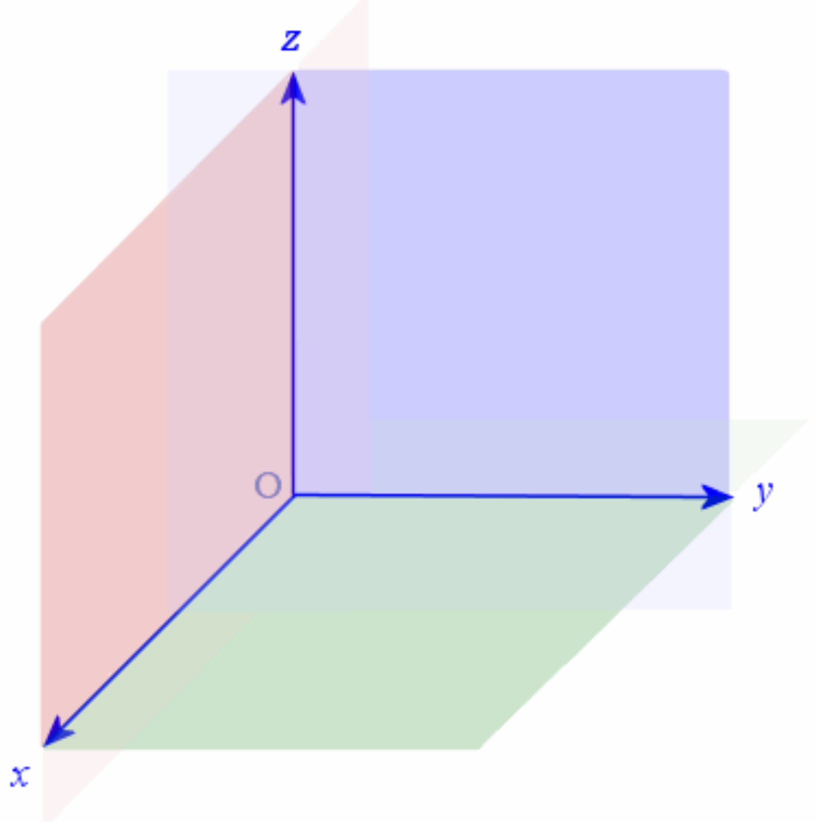
- 什么复杂度？$X^\top X$需要$O(nd^2)$，是主要复杂度项（另外，求逆需要$O(d^3)$）

关于$X^\top X$的可逆性

也就是任何一列都不能表示成其他列的线性组合

- $X^\top X$可逆当且仅当X的rank是d，也就是X的d列线性独立
- 对于linear regression我们一般假定X的列都是独立的
  - 不独立的话，可以只保留独立的列
  - 那些不独立的列本来也不影响x和y之间的线性关系
- 另一种操作（会引入微小误差）：将X每一个元素加上微小均匀随机噪声
  - 噪声范围足够小即可保证对结果影响不大

* 随机矩阵的rank



设d个列对应的向量为$m_1, \dots, m_d \in \mathbb{R}^n$

- 考虑一个$n \times d$随机矩阵M，其中每个M_{ij}都是一个独立的$[0,1]$上均匀随机数
- 任取M的一列，例如第一列m_1，则该列可被其他列线性表出的概率是0
  - 由于m_1独立于其他m_i，可假定其他m_i都确定后再选取m_1
  - 考虑所有的$m_2, \dots, m_d$的线性组合$\{\lambda_2 m_2 + \dots + \lambda_d m_d : \lambda_2, \dots, \lambda_d \in [0,1]\}$
  - 这些线性组合落在单位方$[0,1]^d$的一个$(d-1)$维子空间，而m_1可以取遍整个方
  - 这样m_1落在$d-1$维面上的概率就是0，也就是可被线性表出的概率是0

d维方的$d-1$维面/子空间的体积是0

优化复杂度：降维

回顾：目标是$\arg\min_{w\in\mathbb{R}^d} \|Xw - y\|^2$

- 这里Xw和y都是n维的向量，最后目标是求这两个向量的距离（的平方）
- 能否用一个JL矩阵$A \in \mathbb{R}^{m\times n}$，得到m维的$AXw$和$Ay$，使得

$$\|AXw - Ay\|^2 \in (1 \pm \epsilon)\|Xw - y\|^2$$

m是target dimension

- 如果这样可行，那么就预处理AX和Ay后解regression，可以把n降成m

因此刚刚的$O(nd^2)$复杂度变为$O(md^2)$

问题是?

- 但：我们到底是对哪些向量保范数平方呢？这决定了m该取多少

也就是先降维，然后在低维上解问题

- 目标：$\arg\min_{w\in\mathbb{R}^d} \|Xw-y\|^2$，且我们想$\arg\min_{w'\in\mathbb{R}^m} \|AXw'-Ay\|^2$只有$1\pm\epsilon$误差
- 这要求：对所有$w\in\mathbb{R}^d$都保$\|AXw-Ay\|^2$，而不仅仅是最优解w^*
- 回忆JL保证：

否则，降维后的代价$\|AXw-Ay\|^2$对某个w甚至可以远小于真实的$\|Xw-y\|^2$，甚至小于最优解！这会让整个优化问题完全失去意义

$$\forall v\in\mathbb{R}^d,\qquad \Pr[\|f(v)\|^2\in(1\pm\epsilon)\cdot\|v\|^2]\geq 1-\exp(-O(\epsilon^2 m))$$

理论上需要对所有、无穷多个$v=Xw-y$应用JL

此时用union bound的话，target dimension m需要是无穷？！

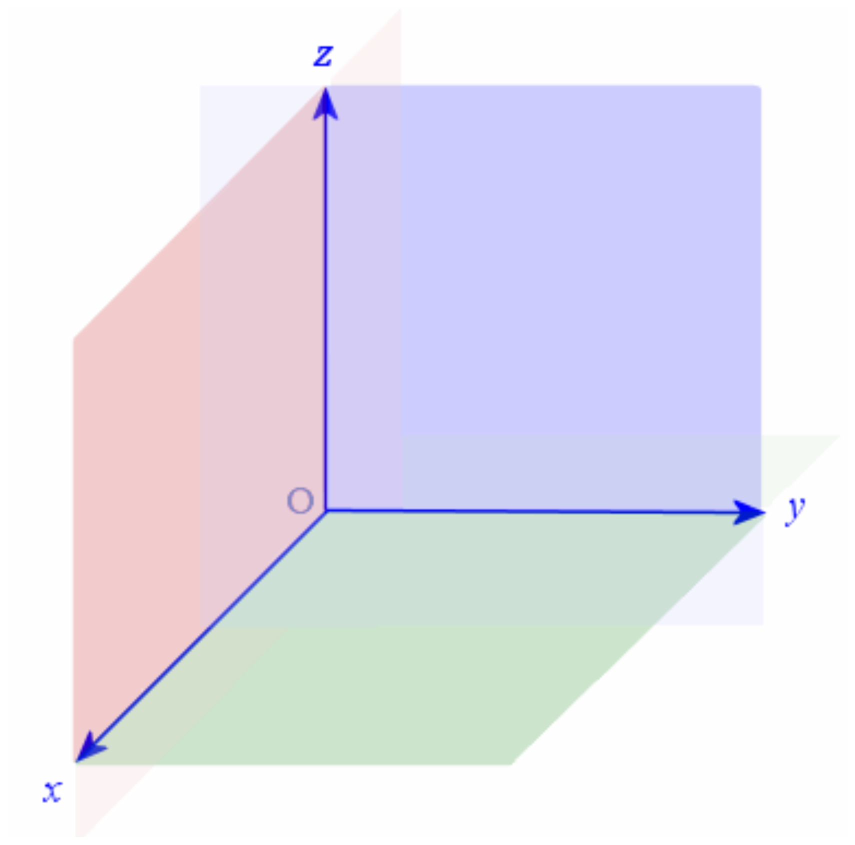
解决方案：Subspace版本的JL

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$\mathcal{U}$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



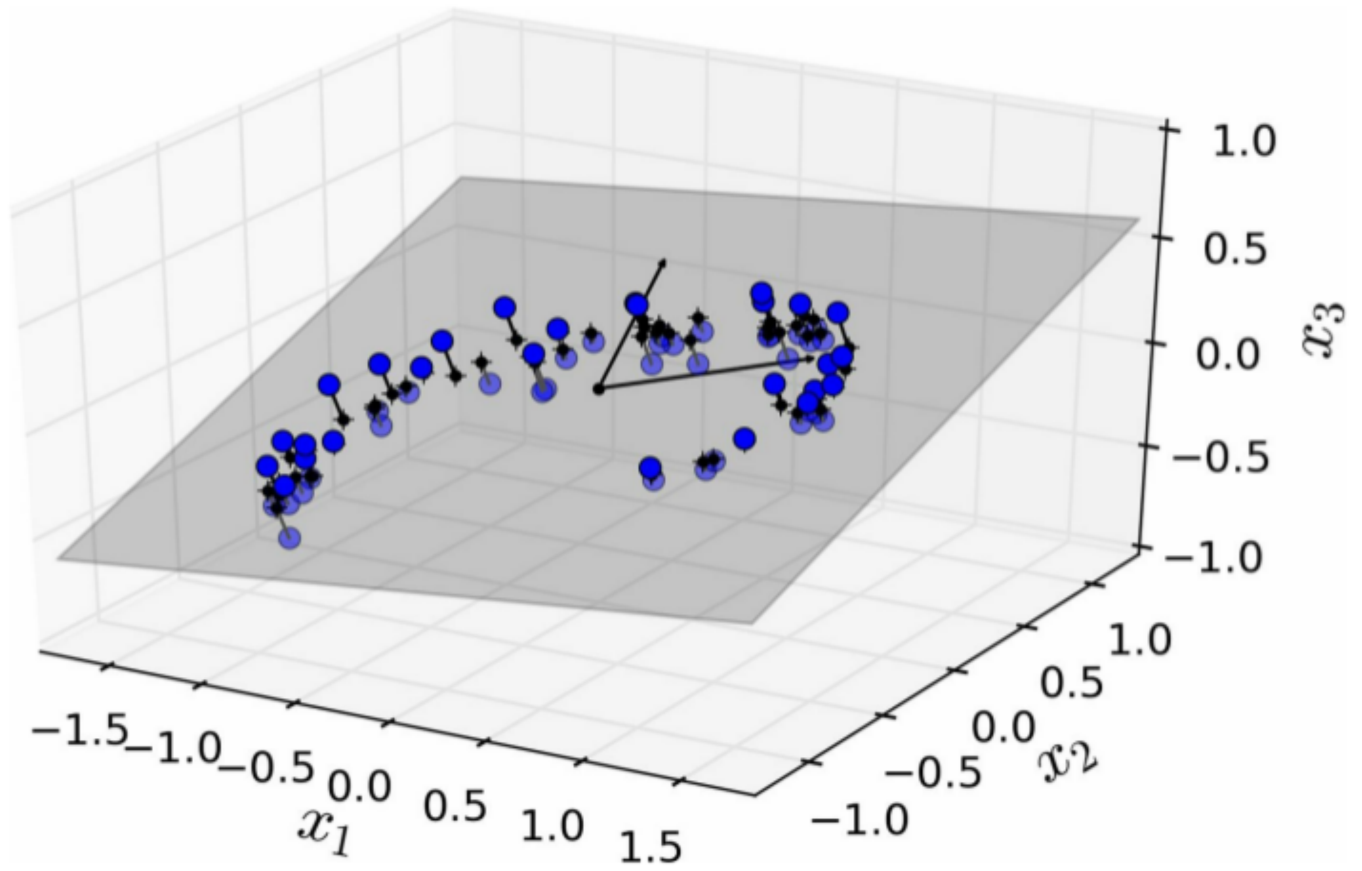


Figure 8-2. A 3D dataset lying close to a 2D subspace

$Xw - y$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$m = O(d)$

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

- 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！

进一步优化

- 注意到需要预先对X用JL，即计算AX
  - A大约$d \times n$维，X是$n \times d$维，依然需要nd^2，与原来的$X^\top X$比也没更好……
- 另外：一般而言X都是稀疏的，即多数位置都是0
  - 记$\mathrm{nnz}(X)$为X中非0元素的个数，nnz = number of non-zero

因此，我们想要$O(\mathrm{nnz}(X))$的复杂度来计算AX!

$O(\mathrm{nnz}(X))$时间降维

设随机$S \in \mathbb{R}^{m\times n}$每列只有一个均匀随机位置非0，且该元素取均匀随机$\{-1,1\}$

$$S = \begin{pmatrix} 0 & 0 & \cdots & 1 & 0 & -1 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 1 & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & 1 & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 0 \end{pmatrix}$$

下页解释SX可在$O(\mathrm{nnz}(X))$计算！

[Clarkson-Woodruff, STOC 13]

定理：用$A' = S$替代JL矩阵A，$m = O(\epsilon^{-2}d^2 \cdot \mathrm{poly}\log(\epsilon^{-1}d))$有类似保证

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

d^2会让最后降维后的矩阵维度稍大，后面会讨论

回忆$\mathcal{U}$是$\mathbb{R}^n$的d维子空间

为什么SX可在$O(\mathrm{nnz}(X))$时间计算?

- 因为要利用X的稀疏性，我们考虑某个非零的X_{ij}如何影响结果SX
  - 首先只会影响SX的第j列的某（些）行；但根据S性质，只有一行会被影响
  - 具体：设S的第i列选取的随机行是$\pi(i)$，那么X_{ij}只会影响$(\pi(i), j)$
  - 算法：扫描X的非0项X_{ij}，更新结果矩阵$M_{\pi(i),j} \mathrel{+}= X_{ij}$

事实上：即使不考虑稀疏性
$\mathrm{nnz}(X) \le nd$也比nd^2要好

$$S = \left(\begin{array}{c|c|c|c|c|c} 0 & 0 & \cdots & 1 & 0 & -1 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 1 & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & 1 & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 0 \end{array}\right)$$

这种稀疏矩阵S目标维度m关于d的依赖的讨论

- 由于S只能做到$m = O(d^2)$，降维后得到的矩阵是$d^2 \times d$的
  - 这样后续求解依然需要$O(d^4)$时间，而不是subspace JL能得到的$O(d^3)$
- 相对于subspace JL的优点：降维复杂度$\mathrm{nnz}(X)$，JL是nd^2

为什么需要d^2

- 考虑$n = d$并且rank = d的数据矩阵X
  - 则：$\min_w \|Xw - y\|^2 = 0$（即直接解方程$Xw = y$）

$$S = \left(\begin{array}{c|c|c|c|c|c} 0 & 0 & \cdots & 1 & 0 & -1 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 1 & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & 1 & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 0 \end{array}\right)$$

- 此时SX需要rank = d才能保证SX上的最优值 = 0
- 回忆：S每列恰取1行填入非0元，则$\mathrm{rank}(S) = d$仅当S的列选取至少d个不同行
  - 等价于每次选取的随机行都不同，概率 $\approx \frac{m-1}{m} \cdot \ldots \cdot \frac{m-d+1}{m}$

$$
\begin{aligned}
&\frac{m-1}{m}\cdot\ldots\cdot\frac{m-d+1}{m} \\
&= (1-1/m)\cdot(1-(d-1)/m) \\
&\approx \exp(-\sum_{j=0}^{d-1} j/m) \\
&\approx \exp(-d^2/m)
\end{aligned}
$$

令这个概率等于一个常数，那么$m \geq \Omega(d^2)$

求解linear regression：完整算法

最坏情况$O(\mathrm{nnz}(X) + d^4)$的算法

- 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

* 如何做到$O(\mathrm{nnz}(X)+d^3)$

- 可利用Fast JL进一步将$d^2 \times d$矩阵降维到$d \times d$，运行时间$O(d^3)$
- 回忆：$m=d$的JL矩阵A应用在$n \times d$的矩阵X上复杂性是$O(nd^2)$
- Fast JL: 达到类似JL的保证，但fast JL矩阵A_F应用在X上的时间大约是$O(nd)$

Fast JL：非Subspace版本

原论文：https://www.cs.princeton.edu/~chazelle/pubs/FJLT-sicomp09.pdf

- 给定n个在$\mathbb{R}^d$的数据点集合X

与传统JL同阶

- 结论：存在$m \times d$降维矩阵A_F，$m = O(\epsilon^{-2} \log n)$，使得
  - $\Pr[\forall x \in X, \|A_F x\| \in (1 \pm \epsilon)\|x\|] \geq 2/3$
  - $\forall x \in X$，$A_F x$可在$\tilde{O}(m + d)$时间计算

传统JL需要$O(md)$

$\tilde{O}$隐藏poly $\log(mnd)$

- 所以：对所有n个点做降维，总复杂度是大约是$n(m + d)$

Fast JL：非Subspace版本构造

- $m \times d$的降维矩阵$A_F := (md)^{-1/2} \cdot PHD$
  - D是$d \times d$的对角阵，每个对角线上元素以0.5概率取+1，另0.5概率取-1
  - H是$d \times d$的Hadamard矩阵，$H_{ij} := (-1)^{\langle i-1, j-1 \rangle}$，其中$\langle i, j \rangle$是把$i$和$j$的二进制表示向量做内积 $H_{2^t} = \begin{bmatrix} H_{2^{t-1}} & H_{2^{t-1}} \\ H_{2^{t-1}} & -H_{2^{t-1}} \end{bmatrix}$, $H_1 = [1]$

    Hy可以快速计算（并不直接；需要特别设计）
  - P是$m \times d$矩阵，每个元素是$B(q) \cdot N(0, q^{-1})$，其中$B(q)$是取1的概率为q的{0, 1}分布，$N(0, q^{-1})$是期望0方差q^{-1}的正态，$q = \min\{\Theta(\log^2(n)/d), 1\}$

Fast JL：Subspace版本

- 给定$n \times d$的数据矩阵X
- fast JL给出一个$m \times n$的A_F，其中$m = O_\epsilon(d)$，使得
  - 满足subspace保证，即$\Pr[\forall v \in \mathcal{U}, \|A_F v\|_2 \in (1 \pm \epsilon)\|v\|]$
  - 设X的d列是$X_1, \ldots, X_d$，则$A_F X_i$都可以在$\tilde{O}(m+n)$时间计算
- 总共可以在大约$O(d(n+m)) \approx nd$时间计算出$A_F X$

与传统subspace JL同阶

因为$A_F X = [A_F X_1, \ldots, A_F X_d]$

Fast JL：Subspace版本构造

- 与非subspace版本基本相同，只是参数稍微改一下
- $m \times n$的降维矩阵$A_F := (mn)^{-1/2} \cdot PHD$
  - D是$n \times n$的对角阵，每个对角线上元素以0.5概率取+1，另0.5概率取-1
  - H是$n \times n$的Hadamard矩阵，$H_{ij} := (-1)^{\langle i-1, j-1 \rangle}$，其中$\langle i, j \rangle$是把$i$和$j$的二进制表示向量做内积
  - P是$m \times n$矩阵，每个元素是$B(q) \cdot N(0, q^{-1})$，其中$B(q)$是取1的概率为q的{0, 1}分布，$N(0, q^{-1})$是期望0方差q^{-1}的正态，$q = \min\{\Theta(\log^2(n)/n), 1\}$

几种降维方法对Linear Regression的效果总结

方法	目标维度	降维所需时间	降维后求解时间
Subspace JL	$m = O(d)$	$O(nmd)$	$O(d^3)$
Fast JL	$m = O(d)$	$O(d(n+m))$	$O(d^3)$
Sparse S	$m = O(d^2)$	$O(\mathrm{nnz}(X))$	$O(d^4)$
Sparse S + Fast JL	$m = O(d)$	$O(\mathrm{nnz}(X) + d^3)$	$O(d^3)$

先用Sparse S在$\mathrm{nnz}(X)$时间降维到d^2维（对应于新的点数）
再用Fast JL在$d(n+m) = d(d^2+d)$时间降维到d维，最后用d^3公式求解

* 如何证明基础版本Subspace JL?

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：

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

- 简化：只需要对单位模长的v证明即可（因为范数和A都是线性操作）
- 这样的话就只需要证$\|Av\| \in (1 \pm \epsilon)$大概率成立

模长可以从$\|Av\|^2$和$\|v\|^2$提取出来约掉

总思路

- 黑盒使用JL结论：

JL核心结论：

$$\forall v \in \mathbb{R}^d, \qquad \Pr[\|f(v)\|^2 \in (1 \pm \epsilon) \cdot \|v\|^2] \geq 1 - \exp(-O(\epsilon^2 m))$$

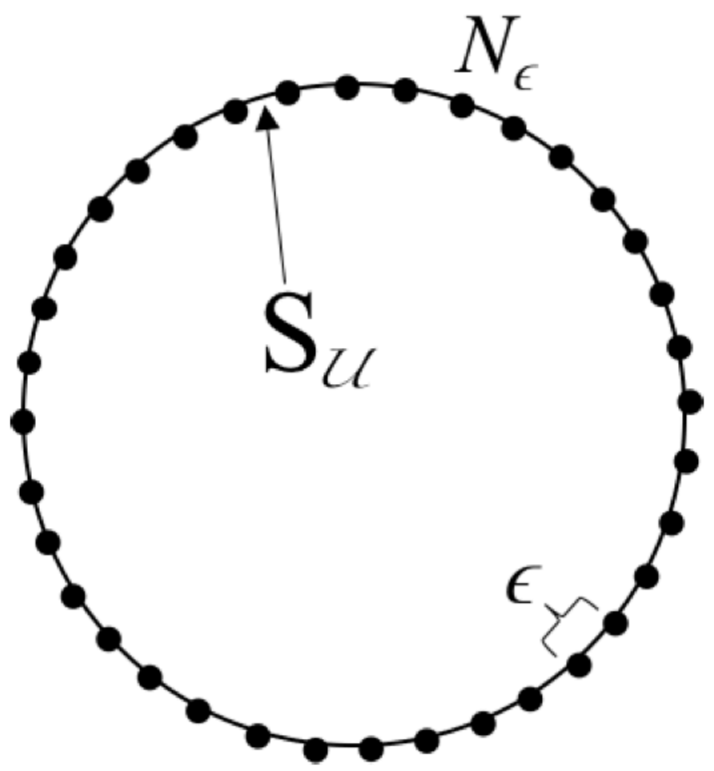
- 直接union bound的问题：需要在整个单位球上的无穷个点上应用，$m \to \infty$...
- 因此需要离散化，使得只需要保有限点上的模长，就可以保所有单位球上的

* 离散化：ϵ-net

- ϵ-net：d维单位球面$\mathcal{S}_{\mathcal{U}}$上的离散子点集$N_\epsilon$，使得
  - (covering) 任何球面点x都能找到一个在N_ϵ中距离 $\leq \epsilon$的点
  - (packing) 任何两个N_ϵ中的点距离 $\geq \epsilon$
- ϵ-net显然是存在的
  - 贪心：从球面$\mathcal{S}_{\mathcal{U}}$任取一点$u$，去掉$B(u, \epsilon)$，在剩下的球面上重复该过程

以u为中心，半径是ϵ的区域/球

2维子空间的ϵ-net



- 定理：任何ϵ-net N_ϵ的大小至多是$O(\epsilon^{-d})$

可以用体积来论证：N_ϵ每个点为中心$\epsilon/2$半径球互不相交
所有球包含在$O(1)$半径的一个大球内；最多允许有多少小球?

* 完成证明：用ϵ-net表示v、并对ϵ-net用JL

该引理的证明稍后给出

引理：考虑一个$v \in \mathcal{S}_{\mathcal{U}}$以及此时可以把$v$表示成

$$v = x_0 + c_1 x_1 + \ldots,\ \text{其中} x_i \in N_\epsilon,\ \text{使得} |c_i| \leq \epsilon^i$$

JL性质：设A是JL矩阵，并且假定A同时保持了N_ϵ内所有点的范数，即

$$\forall u \in N_\epsilon,\ \|Au\|_2 \in (1 \pm \epsilon) \cdot \|u\|_2 \in (1 \pm \epsilon)$$

由于$|N_\epsilon| \leq \epsilon^{-d}$，target dimension只需要是$O(\epsilon^{-2}\log(\epsilon^{-d}))$

则对任意的$v \in \mathcal{S}_{\mathcal{U}}$：

$$\begin{aligned}\|Av\| &= \|Ax_0 + c_1 Ax_1 + \ldots\| \\ &\leq \|Ax_0\| + c_1\|Ax_1\| + \ldots \\ &\leq (1+\epsilon) + (1+\epsilon) \cdot \sum_{j=1}^{\infty} \epsilon^j \leq 1 + O(\epsilon)\end{aligned}$$

三角形不等式

利用$\|Ax_i\| \leq (1+\epsilon)$

另一个方向依然可用三角形不等式：
$\|Ax_0 + c_1 Ax_1 + \ldots\|$
$\geq \|Ax_0\| - c_1\|Ax_1\| - c_2\|Ax_2\| - \ldots$

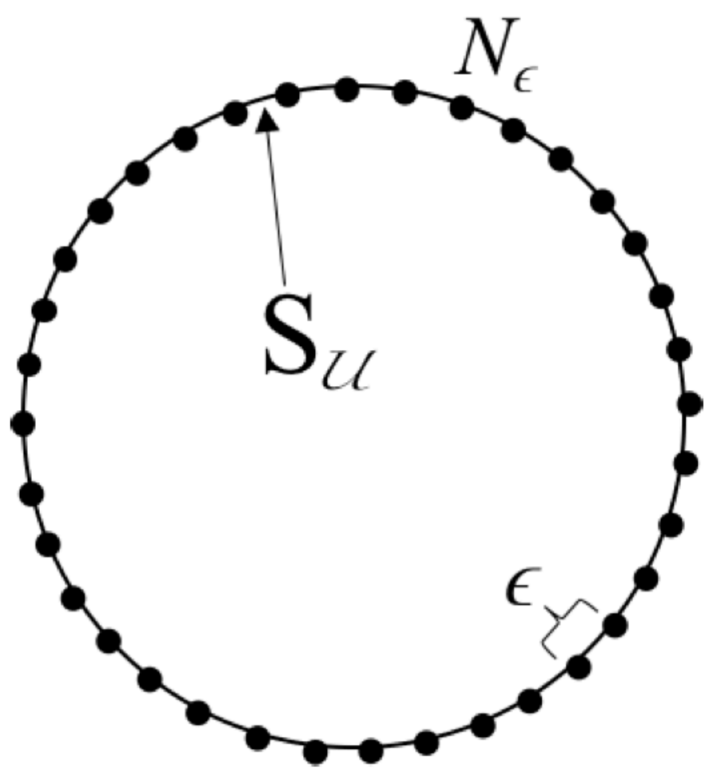
* 证明引理：如何用ϵ-net表示v

引理：考虑一个$v \in \mathcal{S}_{\mathcal{U}}$，此时可以把$v$表示成

$v = x_0 + c_1x_1 + \dots$，其中$x_i \in N_\epsilon$，使得$|c_i| \leq \epsilon^i$

可以是一个无穷项的求和

2维子空间的ϵ-net



证明引理：取x_0为v在N_ϵ中距离ϵ以内的点

- 故$v - x_0$是一个norm至多是ϵ的向量

所以$\|\frac{v - x_0}{\|v - x_0\|} - x_1\| \leq \epsilon$，推出$\|v - x_0 - c_1x_1\| \leq \epsilon c_1 \leq \epsilon^2$

- 定义x_1为$\frac{v - x_0}{\|v - x_0\|_2}$在$N_\epsilon$中距离$\epsilon$内的点，取$c_1 = \|v - x_0\|_2 \leq \epsilon$

需要归一化才能用net的性质

- 递归对$v - x_0 - c_1x_1$进行操作，得到$c_2 \leq \epsilon^2$，且$\|v - x_0 - c_1x_1 - c_2x_2\| \leq \epsilon^3$

之后依此类推

Kernel的降维：RFF方法

Kernel

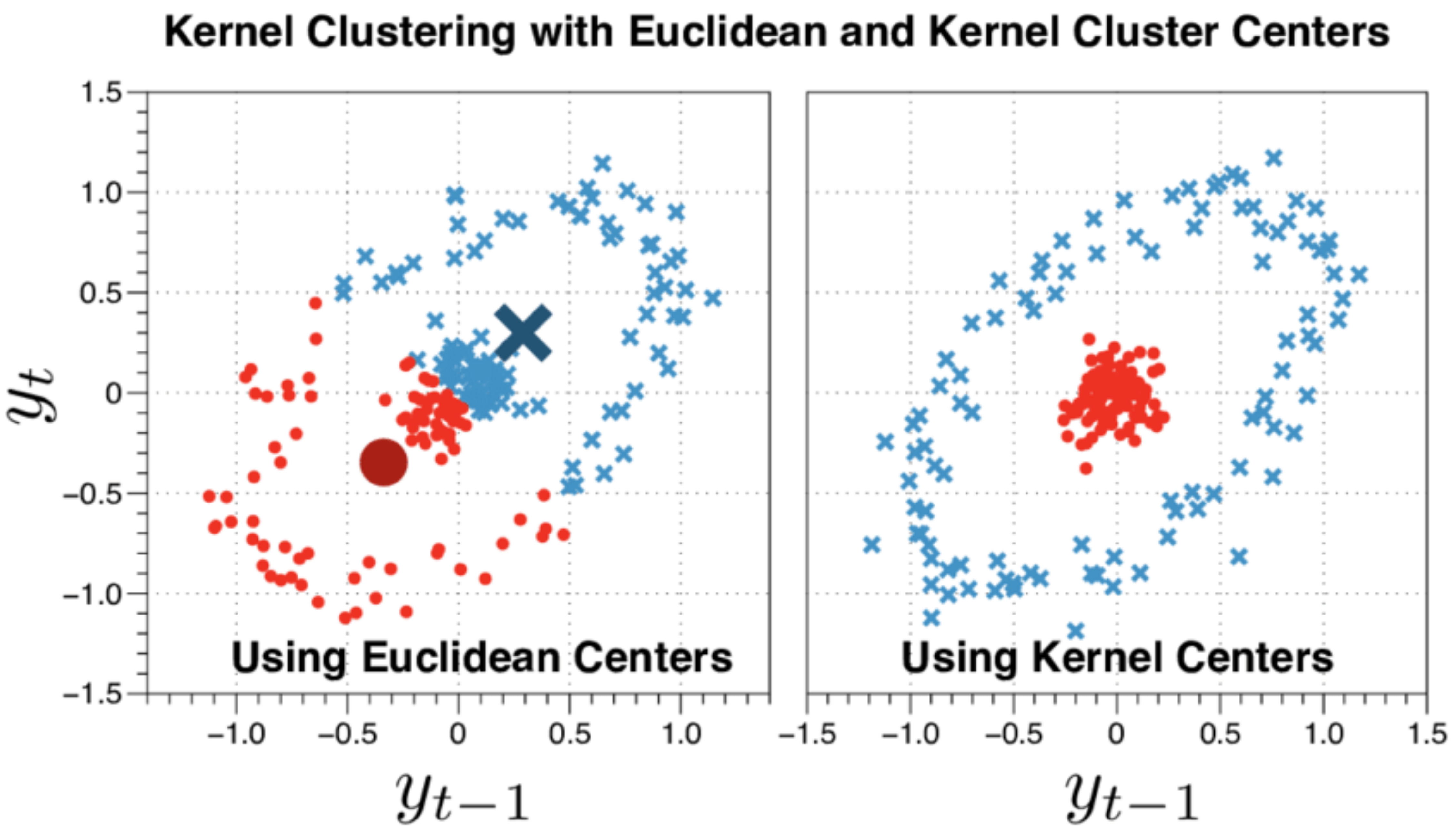
- P是n个$\mathbb{R}^d$上的数据点
- kernel K是一个函数$K : P \times P$
  - 函数值满足：存在一个$\varphi : \mathbb{R}^d \to \mathbb{R}^n$使得$\forall x, y \in P$, $K(x, y) = \langle \varphi(x), \varphi(y) \rangle$
  - 即通过K定义的一种高维的cosine similarity
- 著名例子：Gaussian kernel K_G

一般而言可以是无穷维，但仅对n个数据点来说，
则存在一个n维的表示

$$K_G(x, y) := \exp(-\|x - y\|_2^2 / c)$$

Kernel的一般用途

- 在kernel定义的$\varphi(P)$上做计算，而不是在原点集P上



Kernel的计算挑战

- 定理：给出所有的$K(x,y)$，可以在$\mathrm{poly}(n)$时间内计算出所有n个点的$\varphi(\cdot)$
  - 但这个poly至少有n^4
- 但有时候可以间接计算，例如

$$\|\varphi(x)-\varphi(y)\| = \sqrt{\|\varphi(x)-\varphi(y)\|^2} = \sqrt{K(x,x)+K(y,y)-2K(x,y)}$$

- 然而很多快速算法还是需要直接得到$\varphi(\cdot)$的坐标表示才能运行

Random Fourier Features (RFF)

[Rahimi & Recht, NeurIPS 2007]

输入空间，不是kernel的φ空间

- 类似于JL，想要定义一个$f : \mathbb{R}^d \to \mathbb{R}^m$

m是目标维度

- RFF方法：对于shift-invariant kernel，即满足$K(x, y) = K(x - y)$的K，定义

$$\pi(x) := \sqrt{\frac{1}{m}} \begin{pmatrix} \sin\langle\omega_1, x\rangle \\ \cos\langle\omega_1, x\rangle \\ \vdots \\ \sin\langle\omega_m, x\rangle \\ \cos\langle\omega_m, x\rangle \end{pmatrix}$$

，其中ω_i是iid的，对于Gaussian kernel应取$\mathcal{N}(0,1)$

对于其他kernel的具体取法参考[RR07]
基本上需要求一个K上的傅立叶变换类型的积分

$$p(\omega) = \frac{1}{2\pi} \int_{\mathbb{R}^d} K(x) e^{-i\langle\omega, x\rangle}$$

RFF的保证

对任何shift-invariant kernel成立

- [RR07]的结论：$\mathbb{E}[\langle \pi(x), \pi(y) \rangle] = K(x-y)$
  - 至少是对kernel function的无偏估计
  - 足够大的target dimension一定可以良好的近似

仅对Gussian kernel等“可解析”的shift-invariant kernel 成立

- 我们的结论：$m = \max\{\epsilon^{-1}\log^3 n, \epsilon^{-2}\log n\}$，可保pairwise kernel distance

$$\forall x, y \in P, \quad \|\pi(x) - \pi(y)\| \in (1 \pm \epsilon) \cdot \|\varphi(x) - \varphi(y)\|$$

[Cheng-**J**-Wei-Wei, ICLR 2023]

实际应用时：类似JL，设置一个目标的m，然后测试误差，平均来看效果良好

RFF总结

- RFF是data oblivious的，类似于JL
  - 其他对kernel的降维，例如所谓的Nystrom method，不是data oblivious的
- RFF的主要用处在于避免暴力求解φ
  - 如果允许求解φ，则可以在φ空间上直接利用JL
- RFF只适用于shift-invariant kernel，但这已经包括Gaussian kernel等流行kernel