程序设计实习（实验班-2024春）
降维：PCA方法

授课教师：姜少峰
助教：冯施源 吴天意

Email: shaofeng.jiang@pku.edu.cn

PCA降维：
揭示数据维度间的隐含依赖关系

一个引例

	原神	Splatoon	PUBG Mobile	塞尔达传说
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

- 与线性回归不同：无label y，只是一些向量
- 有什么内在关系?
- 是否可能将这个4维数据以某种方式在2维平面展现出来，体现内在关系?

一种新的表达

	原神	Splatoon	PUBG Mobile	塞尔达传说
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

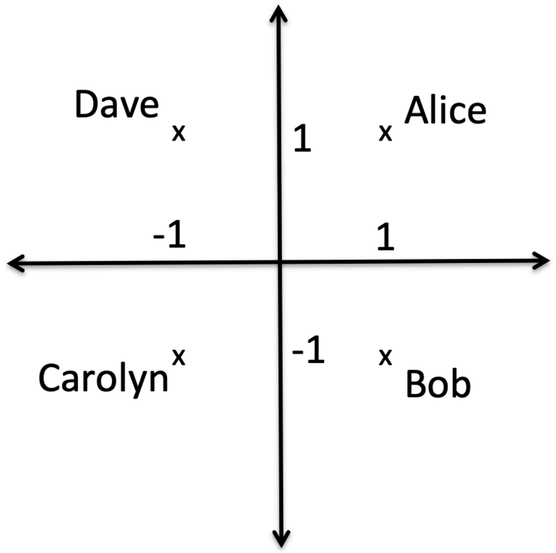
平均值向量

- $\bar{\mathbf{x}} = \frac{1}{4}(\mathbf{x}_1 + \ldots \mathbf{x}_4) = (5.5,4.5,5,5.5)$
- 所有向量可以近似表示为$\mathbf{x}_i = \bar{\mathbf{x}} + a_i\mathbf{v}_1 + b_i\mathbf{v}_2$
  - $\mathbf{v}_1 = (3,-3,-3,3),\ \mathbf{v}_2 = (1,-1,1,-1)$
- A = (1, 1), B = (1, -1), C = (-1, -1), D = (-1, 1)

A代入得到$(9.5,0.5,3,7.5)$，近似相等

利用新表达进行可视化

- $\mathbf{x}_i = \bar{\mathbf{x}} + a_i\mathbf{v}_1 + b_i\mathbf{v}_2$在这种表达中，$(a_i, b_i)$可以看作是新的“坐标”
- A = (1, 1), B = (1, -1), C = (-1, -1), D = (-1, 1)
- 若有更多的点，可根据相对ABCD的远近来定义类别/聚类



低维表达的意义

	原神	Splatoon	PUBG Mobile	塞尔达传说
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

- $\mathbf{x}_i = \bar{\mathbf{x}} + a_i\mathbf{v}_1 + b_i\mathbf{v}_2$在这种表达中，$(a_i, b_i)$可以看作是新的“坐标”
  - 那么$\mathbf{v}_1 = (3, -3, -3, 3)$，$\mathbf{v}_2 = (1, -1, 1, -1)$就是“坐标轴”/“基”
- 如何理解$\mathbf{v}_1$和$\mathbf{v}_2$?

	原神	Splatoon	PUBG Mobile	塞尔达传说
Alice	10	1	2	7
Bob	7	2	1	10
Carolyn	2	9	7	3
Dave	3	6	10	2

- $\mathbf{v}_1 = (3, -3, -3, 3)$，$\mathbf{v}_2 = (1, -1, 1, -1)$
  - 在$\mathbf{v}_1$中第1、4维度是正数，认为是“正相关”
  - 2、3维度是负数，认为他们之间“正相关”，但与1、4之间都是负相关
- 1、4都是动作游戏，2、3是射击，可将$\mathbf{v}_1$系数a_i看作某人喜欢动作游戏程度
- 对于$\mathbf{v}_2$：可看作喜欢手机游戏的程度，但$\mathbf{v}_2$绝对值较低，因此不如$\mathbf{v}_1$“显著”

PCA

中文 = 主成分

- $\mathbf{v}_1$和$\mathbf{v}_2$就是数据集所谓的top 2 principle components
- PCA就是系统性定义、寻找哪些是principle components

与JL的对比

- JL保欧氏距离，PCA一般没法保
- JL是data oblivious，而PCA正相反，找到的基是数据特殊内部结构的反映
- JL的m个随机高斯也可以理解成某种“基”，但没有任何实际意义
  - 而PCA的基经常都对应明显的实际意义
- PCA的target dimension即使等于1、2也是有意义的，而JL需要更高才能保距离

PCA定义

PCA：大体目标

- 输入：n个d维数据点$\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- 想找到m个d维“基”$\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^d$，使得每个$\vec{x}_i$可以用这些基近似表示

m类似JL的target dimension
一般$m \ll n$，甚至为了可视化，经常取$m = 2$

$$\forall 1 \le i \le n, \quad \mathbf{x}_i \approx \sum_{j=1}^{m} a_{ij}\mathbf{v}_j$$

- 刚刚的例子n = 4, d = 4, m = 2

所谓用“基”来表示，意思就是用基的线性组合来表示

- PCA的定义就是要规定怎样的基$\mathbf{v}_1, \ldots, \mathbf{v}_m$是“最佳近似”

通过规定某种目标函数

类似于linear regression，但……

linear regression要找$\mathbf{w}$使得$\mathbf{y}_i \approx \langle \mathbf{w}, \mathbf{x}_i \rangle$

刚刚提到的PCA的目标：$\mathbf{x}_i \approx \sum_{j=1}^{m} a_{ij}\mathbf{v}_j$

也是线性关系；如果$m = 1$那么就也是找一条拟合直线

都是用m维（低维）来近似，或者说“拟合”

- linear regression拟合的是x和y
- PCA拟合的是自身

必要的预处理

均值归0

严格来说不预处理也可以通过修改定义来解决，但那样引入了很多不必要的麻烦

- 对于PCA来说需要进行如下预处理才能良定义
- 均值$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$需要是0：可先计算点集的$\bar{\mathbf{x}}$，然后更新$\mathbf{x}_i := \mathbf{x}_i - \bar{\mathbf{x}}$
  - 当找到m个“基”之后，可以再加上$\bar{\mathbf{x}}$平移回去

必要的预处理

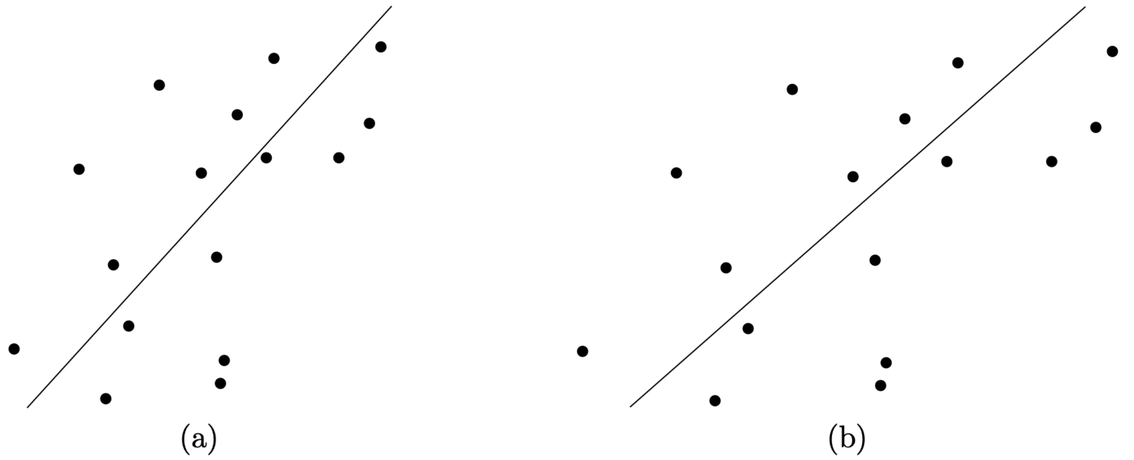
去标度化

标准的PCA定义不是标度无关的

- 数据点的每一维可能代表不同的“单位”，这会干扰最终结果
  - 例如m = 1，那么最终结果就是一条“拟合”直线
  - 如果数据点在某个维度上进行了拉伸，那么这条线也会跟着改变

直线是m = 1维的
xy面是m = 2维的…

例如将单位从“千米”换成了“米”



(a) (b)

去标度：列/维度归一化

- 我们想要一个“标度”无关的结果，因此一般把每一维/列的ℓ_2范数归一化，即

$$\forall 1 \le i \le n, 1 \le j \le d, \quad \mathbf{x}_{ij} := \mathbf{x}_{ij} / \sqrt{\sum_{i'=1}^{n} \mathbf{x}_{i'j}^2}$$

例如：$\begin{bmatrix} 1 & 10 & 5 & 7 \\ 2 & 3 & 4 & 8 \\ 2 & 5 & 1 & 9 \end{bmatrix}$ 第一列范数是3，归一化后是$(1/3, 2/3, 2/3)$；其他列可类似操作

目标函数

- 先考虑target dimension $m = 1$的情况，PCA要找一个最佳“拟合”数据的直线

$$\arg \min_{\mathbf{v}:\|\mathbf{v}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\text{dist between } \mathbf{x} \text{ and line spanned by } \mathbf{v})^2$$

直观上类似linear regression

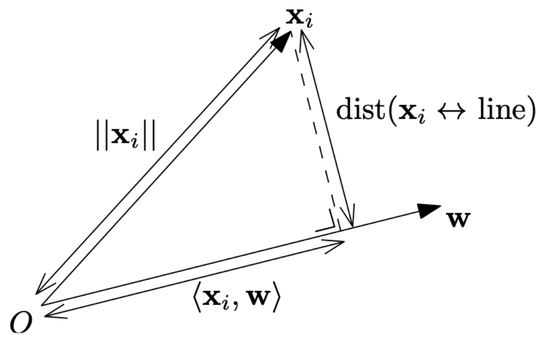
要找一个方向$\mathbf{v}$

line spanned by $\mathbf{v}$就是以$\mathbf{v}$为方向过原点的直线

即点到直线距离的平方和

目标函数等价于：最大化variance

- 勾股定理：对任何$\mathbf{x}$和单位向量$\mathbf{w}$有$\text{dist}(\mathbf{x}, \text{line}(\mathbf{w}))^2 + \langle \mathbf{x}, \mathbf{w} \rangle^2 = \|\mathbf{x}\|^2$



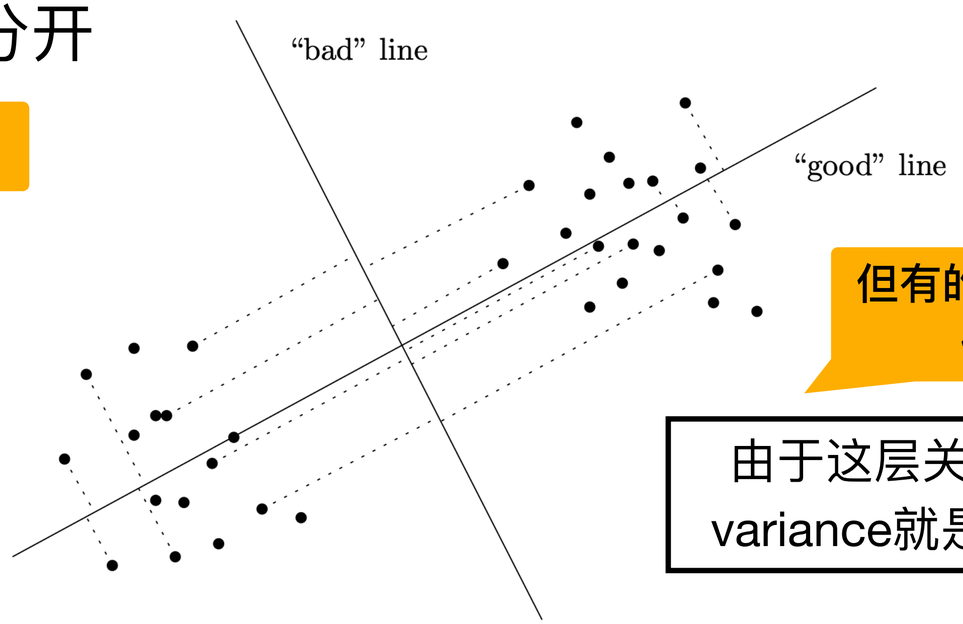
- 因此原目标min点到直线距离平方，就是max投影的长度平方，也就是等价于

$$\arg \max_{\mathbf{v}:\|\mathbf{v}\|_2=1} \frac{1}{n} \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v} \rangle^2$$

这个求和叫做点集投影的variance

- 假若数据有两团/cluster，那么我们想投影到一条线使得这种性质得以保留
  - bad line的问题是投影后在线上来看两团点都混合在一起了
  - good line就还是能分开

对应于最大化variance!



但有的数据中这种variance反而不应该保留，因此PCA可能不适用

由于这层关系，PCA基本上就是认为这种variance就是有用信息，应该尽量得以保留

一般的target dimension m

回忆$m = 1$的情况：

$$\arg \max_{\mathbf{v}:\|\mathbf{v}\|_2=1} \frac{1}{n} \sum_{i=1}^{n} \langle \mathbf{x}_i, v \rangle^2$$

投影长度

一个自然的扩展方法：

$$\arg\min_{m\text{-dim subspace } S} \frac{1}{n} \sum_{i=1}^{n} (\text{length of the projection of } \mathbf{x}_i \text{ on } S)^2$$

可以转化成一个更好“操作”的定义方法

Subspace与Orthonormal Basis

一个m维的subspace可以等价地用m个orthonormal向量的span来表示

- 一组向量$\mathbf{v}_1, \ldots, \mathbf{v}_m$是orthonormal的，若$\|\mathbf{v}_i\| = 1$且$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$ （对$i \neq j$）

normal

orthogonal

$$\text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_m) := \left\{ \sum_{i=1}^{m} \lambda_i \mathbf{v}_i : \lambda_i \in \mathbb{R} \right\}$$

span就是这些向量的所有线性组合

重要性质：$\mathbf{x}$在$\text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_m)$上的投影长度（的平方）等于$\sum_{i=1}^{m} \langle \mathbf{x}, \mathbf{v}_i \rangle^2$

PCA目标函数：最终版本

简化表示

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

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

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

PCA应用

一般框架

- 设置一个m，然后求出top m principle components $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^d$
- 对于某数据点$\mathbf{x} \in \mathbb{R}^d$，$\mathbf{v}_i$-坐标是$\langle \mathbf{x}, \mathbf{v}_i \rangle$，即在$\mathbf{v}_i$上的投影
  - 正数、负数代表什么？绝对值大小代表什么？
- $(\langle \mathbf{x}, \mathbf{v}_1 \rangle, \ldots, \langle \mathbf{x}, \mathbf{v}_m \rangle)$就是$\mathbf{x}$新的$m$维坐标表示
  - 如果m取比较小，可以plot出来

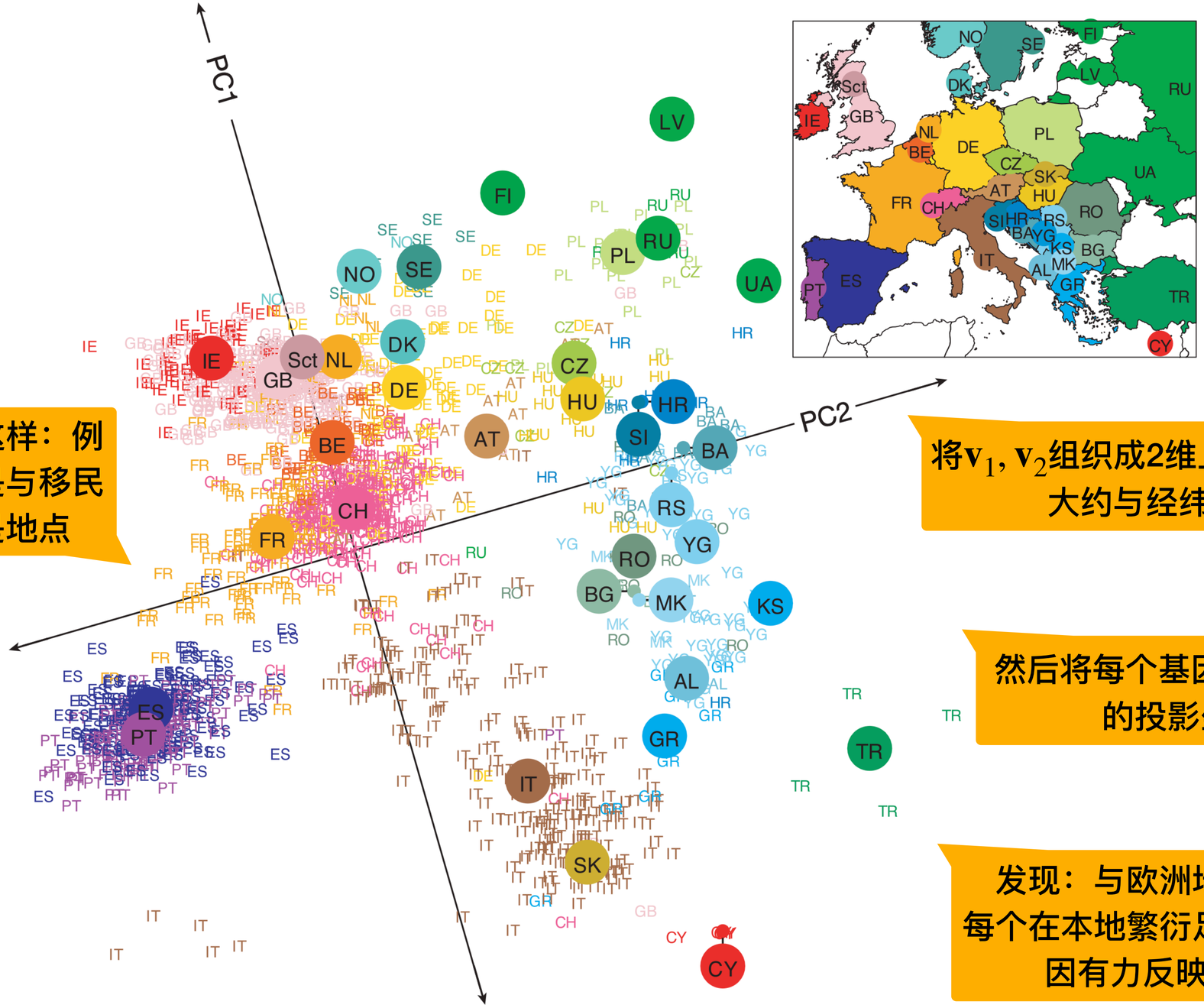
而且我们对任何数据点$\mathbf{x} \approx \sum_{j=1}^{m} \langle \mathbf{x}, \mathbf{v}_j \rangle \mathbf{v}_j$

- 考虑top principle component $\mathbf{v}_1$，如何给$\mathbf{v}_1$一个“物理意义”/“直观解释”?
  - 一般来说，可以看在$\mathbf{v}_1$上最大和最小的坐标及其“附近”的点
  - 这两组点之间的不同点，以及这两组点内部的相同点，可以帮助解释$\mathbf{v}_1$
- $\mathbf{v}_2$一般来说也都能找到一些“物理意义”

PCA应用：基因与地理的联系

Novembre et al., Nature 2008

- 数据集：1387个来自欧洲各地的欧洲人的基因数据
  - 每个数据点是一个200000维的向量，这些维度是基因信息，对基因突变敏感
  - 例如，可能在某个维度上90%人是A，其余10%是C
- 论文采用了$m = 2$的PCA来进行可视化


PC1
PC2
NO
SE
FI
LV
RU
Sct
IE
GB
NL
DE
PL
CZ
SK
UA
FR
CH
AT
HU
SI
HR
RS
RO
BA
YG
KS
BG
IT
MK
AL
GR
PT
ES
TR
CY
DK
BE
在别的地区未必会是这样：例如在美国基因大概率是与移民时间有关系，而不是地点
将$\mathbf{v}_1, \mathbf{v}_2$组织成2维上的垂直向量，大约与经纬线平行
然后将每个基因数据按照$\mathbf{v}_1, \mathbf{v}_2$上的投影坐标画出来
发现：与欧洲地理高度重合！
每个在本地繁衍足够时间的人的基因有力反映了地理位置

PCA应用：Eigenface

- 用PCA来做人脸建模和识别

数据集示例

- 数据集：256 x 256的正面标准人脸数据
  - 展开成65536维的向量，每个维度代表像素值
  - 按照PCA预处理规则，均值归零

男性平均值

减去平均值后的数据

图像“压缩”/“降维”

- 使用PCA，$m \approx 100$，远低于65535，可非常精确地表达这些图像
- 有趣的事实：每个principle component可视化成图像，一般代表某种特征

戴不戴眼镜；留不留胡子etc

这种principle component又叫做eigenface

我们将要讲到principle components是eigenvector

- 逐渐增大m的效果：从平均脸到具体数据

可以在m维上用nearest neighbor search等方法做人脸查询

PCA的局限性?

有时每维归一可能是不够的，也许需要专门设计如何scaling

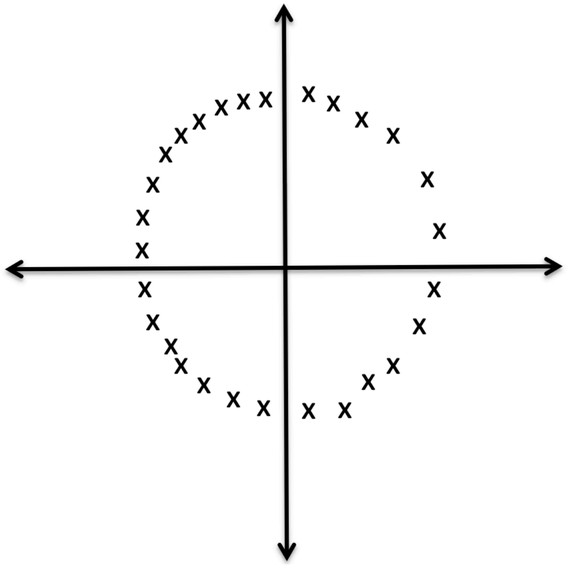
不抗噪声：一个放在无穷远的单点可让解失去意义

PCA类似linear regression，只能找到数据的线性特征

一般只有前几个principle components有物理意义

- 尤其是当数据中没有明显的、多个“orthogonal”意义的时候
- 如欧洲人基因与地理关系的应用中第三个principle component?

非线性、但只有1维/自由度



PCA求解：公式

考虑$m = 1$

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

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \mathbf{x}_i, \mathbf{v}_j \rangle^2$$

$m = 1$的情况：$\arg \max_{\mathbf{v}:\|\mathbf{v}\|=1} \frac{1}{n} \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v} \rangle^2$

重写成矩阵形式

目标函数：$\arg \max_{\mathbf{v}:\|\mathbf{v}\|=1} \frac{1}{n} \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v} \rangle^2$

设$\mathbf{X} \in \mathbb{R}^{n \times d}$为将每个$\mathbf{x}_i$作为一行的矩阵，那么 $\mathbf{X}\mathbf{v} = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{v} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{v} \rangle \end{bmatrix}$

所以目标函数中 $\sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{v} \rangle^2 = \|\mathbf{X}\mathbf{v}\|^2 = (\mathbf{X}\mathbf{v})^\top \mathbf{X}\mathbf{v} = \mathbf{v}^\top \mathbf{X}^\top \mathbf{X}\mathbf{v}$

设$\mathbf{A} = \mathbf{X}^\top \mathbf{X}$，则目标等价于$\arg \max_{\mathbf{v}:\|\mathbf{v}\|=1} \mathbf{v}^\top \mathbf{A}\mathbf{v}$

$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$的意义

covariance matrix

$\mathbf{v}^\top\mathbf{A}\mathbf{v}$又叫做“二次型”
因此这个问题又叫最大化二次型

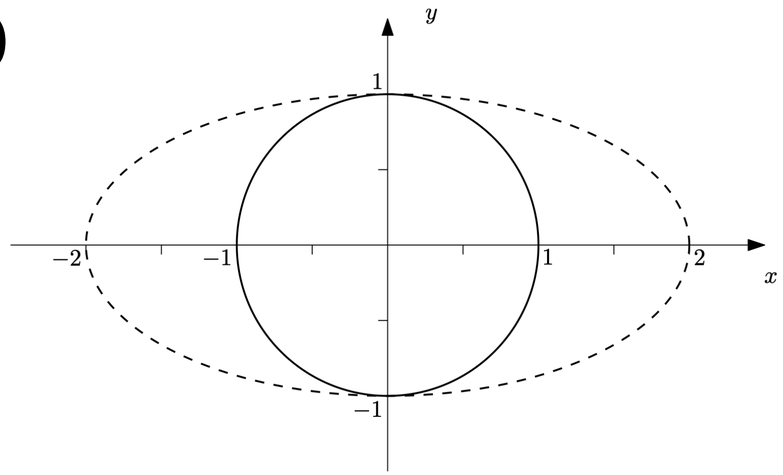
设$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$，则目标等价于$\arg\max_{\mathbf{v}:\|\mathbf{v}\|=1} \mathbf{v}^\top\mathbf{A}\mathbf{v}$

- $\mathbf{A}_{ij}$其实是$\mathbf{X}$的第i列与第j列的点乘
  - 意义：例如$\mathbf{X}$的一列/维代表某个单词在某个文档是否出现（值为0/1）
  - 那么第i列与第j列的点乘就是词i与词j共同出现的文档个数
- $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$叫做covariance matrix

描述correlation

如何求解特殊情况：A是对角阵

特例：$\mathbf{A} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_d \end{bmatrix}$，设$\lambda_1 \geq \ldots \geq \lambda_d \geq 0$



左乘对角阵就是拉伸每个维度，例如$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 2x + y$

一句话：对角阵 = 伸缩变换

对角阵的Principle Component

$$\mathbf{v}^\top(\mathbf{A}\mathbf{v}) = [v_1, \ldots, v_d] \cdot \begin{bmatrix} \lambda_1 v_1 \\ \vdots \\ \lambda_d v_n \end{bmatrix} = \sum_{i=1}^{d} \lambda_i v_i^2$$

取什么$\mathbf{v}$可以最大化?

直观理解：取“拉伸”最大的方向，即$\mathbf{v} = \mathbf{e}_1 = (1,\ldots,0)$

代数上，$\sum_{i=1}^{d} \lambda_i v_i^2$是所有$\lambda_i$的一个加权平均，那么自然应该把权重都放在最大值$\lambda_1$

更进一步：考虑一种“带方向”的伸缩

- 例如右边的变换：

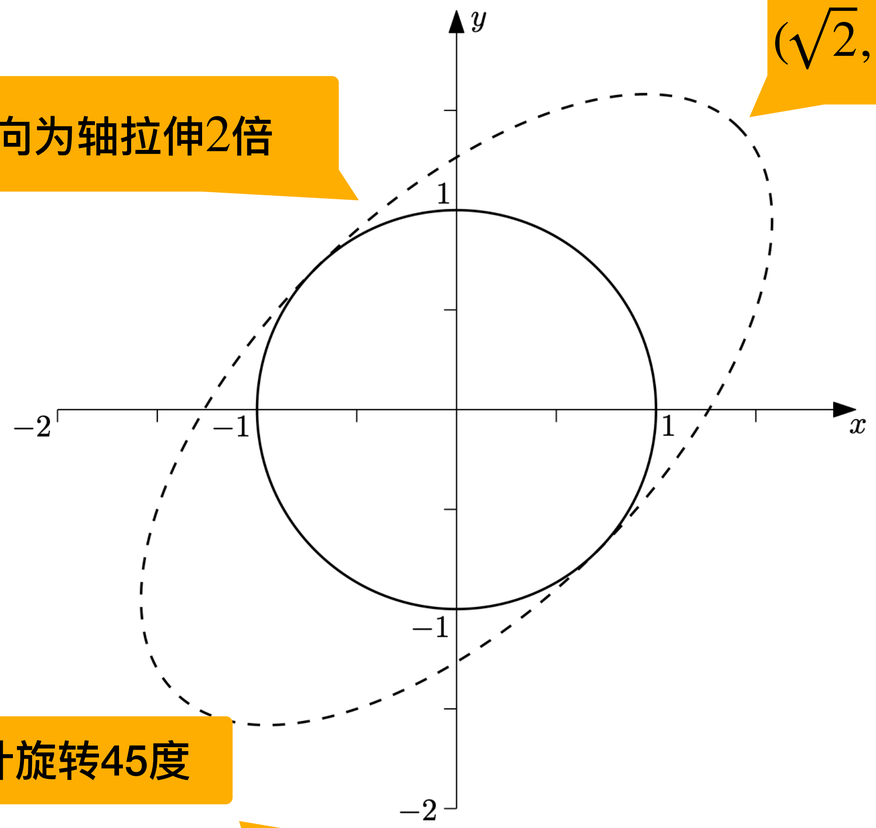
$$\begin{bmatrix} \frac{3}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

最后旋转45度回来

再拉伸2倍

先顺时针旋转45度

将圆沿着45度方向为轴拉伸2倍



这其实是“伪装”的伸缩：
旋转之后伸缩又再转回去

这个旋转矩阵 $\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$ 一般而言是一个orthogonal matrix

Orthogonal Matrix

即一组“基”

定义：$\mathbf{Q}$是orthogonal matrix，若$\mathbf{Q}$是一个方阵且所有列是orthonormal的

性质：

$\mathbf{Q}^\top\mathbf{Q}$的第$i$行$j$列是$\mathbf{Q}$的$i$列与$j$列的点积
然后利用列orthonormal的性质

- $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$：说明$\mathbf{Q}$的逆矩阵就是$\mathbf{Q}^\top$

  这同时也推出了$\mathbf{Q}$的行也是orthonormal的

- 不改变模长：对任何$\mathbf{v}$有$\|\mathbf{Qv}\| = \|\mathbf{v}\|$
  - 因为：$\|\mathbf{Qv}\|^2 = (\mathbf{Qv})^\top(\mathbf{Qv}) = \mathbf{v}^\top\mathbf{Q}^\top\mathbf{Qv} = \mathbf{v}^\top\mathbf{v} = \|\mathbf{v}\|^2$
- $\mathbf{Q}$是旋转矩阵：$\mathbf{Qx}$是$\mathbf{x}$在$\mathbf{Q}$定义的基上投影长度向量，即旋转后的“坐标”

“伪装”的对角阵$\mathbf{A}$：$\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$

- 考虑$\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$，其中$\mathbf{Q}$是orthogonal matrix，$\mathbf{D}$是对角阵
- 这种$\mathbf{A}$是“伪装”的对角阵：先用$\mathbf{Q}^\top$做旋转，用$\mathbf{D}$伸缩，然后$\mathbf{Q}$转回去
- 这种$\mathbf{A}$下最优解是多少?

设$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$，则目标等价于$\arg\max_{\mathbf{v}:\|\mathbf{v}\|=1} \mathbf{v}^\top\mathbf{A}\mathbf{v}$

- 依然应该取拉伸最大的方向，也就是λ_1“对应”的方向
- 单在$\mathbf{D}$上看应是$\mathbf{v} = \mathbf{e}_1$，但$(\mathbf{v}^\top\mathbf{Q})\mathbf{D}(\mathbf{Q}^\top\mathbf{v})$变换后应取$\mathbf{Q}^\top\mathbf{v} = \mathbf{e}_1$即$\mathbf{v} = \mathbf{Q}\mathbf{e}_1$

另一个直觉：带方向的伸缩

这就是$\mathbf{Q}$的第一列

将$\mathbf{Q}^\top\mathbf{v}$看作一个整体，对应$\mathbf{v}$的地位

为什么取$\mathbf{Q}$的第一列：代数解释

- 考虑$\mathbf{A} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$，其中$\mathbf{Q}$是orthogonal matrix，$\mathbf{D}$是对角阵

设$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$，则目标等价于$\arg\max_{\mathbf{v}:\|\mathbf{v}\|=1} \mathbf{v}^\top\mathbf{A}\mathbf{v}$

- 代数上，注意到$(\mathbf{v}^\top\mathbf{Q})\mathbf{D}(\mathbf{Q}^\top\mathbf{v})$依然是对$\lambda_i$的加权平均

注意到$\mathbf{Q}^\top\mathbf{v}$是单位向量，因此是加权“平均”

- 如果代入$\mathbf{v} = \mathbf{Q}\mathbf{e}_1$，得到

$$(\mathbf{v}^\top\mathbf{Q})\mathbf{D}(\mathbf{Q}^\top\mathbf{v}) = (\mathbf{e}_1^\top\mathbf{Q}^\top\mathbf{Q})\mathbf{D}(\mathbf{Q}^\top\mathbf{Q}\mathbf{e}_1) = \mathbf{e}_1^\top\mathbf{D}\mathbf{e}_1 = \lambda_1$$

达到了加权平均的最大可能值，因此是最优的

一般矩阵$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$?

- 事实上我们已经解决了一般情况：
  - 因为$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$必可写成$\mathbf{QDQ}^\top$，且$\mathbf{D}$的每个元素确实都非负
  - 这个命题的具体证明一般在性代数课会涉及
- 因此我们得到了：
  - 对于target dimension $m = 1$，principle component是$\mathbf{Q}$的第一列
  - 这里$\mathbf{Q}$就是$\mathbf{X}^\top\mathbf{X}$的eigendecomposition的$\mathbf{Q}$

叫做eigendecomposition

一般的m?

可以这样考虑：先得到$\mathbf{Q}$第一列当作top principle component，对应的是得到λ_1的方向；再看与它垂直的方向中得到λ_2的就是$\mathbf{Q}$第二列

- 可以用类似的方法得到：top m principle就是$\mathbf{Q}$的前m列
  - 假定$\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$中$\mathbf{D}$是按照从大到小排列的（因此$\mathbf{Q}$的列也据此排列）
  - 可以看到，这m个向量确实是orthonormal的

与eigenvalue/eigenvector的联系

- 回忆什么是特征值/特征向量（eigenvalue/eigenvector）
  - 对矩阵$\mathbf{A}$，满足$\mathbf{Av} = \lambda\mathbf{v}$的$\mathbf{v}$叫eigenvector，$\lambda$叫eigenvalue
- 在$\mathbf{A} = \mathbf{QDQ}^\top$中，$\mathbf{Q}$的每一列$\mathbf{Qe}_i$和$\mathbf{D}_{ii}$分别是第$i$个eigenvector和eigenvalue
  - 因为：$\mathbf{AQe}_i = \mathbf{QDe}_i = \lambda_i\mathbf{Qe}_i$

按特征值从大到小排列的top m

因此PCA等价于：计算$\mathbf{X}^\top\mathbf{X}$的top m eigenvectors

一种常见的PCA的“定义”

关于PCA的“唯一”性

- PCA“基本上”是唯一的
- 首先，$\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$这个分解中，若$\mathbf{D}$从大到小排列，则$\mathbf{D}$可证是唯一的
- 如果$\mathbf{D}$的值都不相等，那么$\mathbf{Q}$也是“唯一”的
- 如果有相等的：考虑一段连续ℓ个相等的$\mathbf{D}$值
  - 对应的$\mathbf{Q}$的ℓ列构成一个ℓ维子空间
  - 任何该子空间的ℓ个orthonormal vectors都可选成对应principle components

事实上每个列向量的+/-都是可能的，但对PCA来说principle component的符号不重要，重要的是张成的subspace/直线

PCA求解：快速近似算法

求解PCA的算法

复杂度较高，基本上是3次方时间，但可以计算出所有principle components

- 大体有两类
- 精确求解：注意到所有的PC都是$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$的特征向量
  - 可以在$O(nd^2)$时间求出所有PC，也就是求出$\mathbf{QDQ}^\top$

这些算法必然是数值（近似）算法，因为求D对应于求解特征多项式的所有根，而五次几以上多项式方程无求根公式

X是n x d的

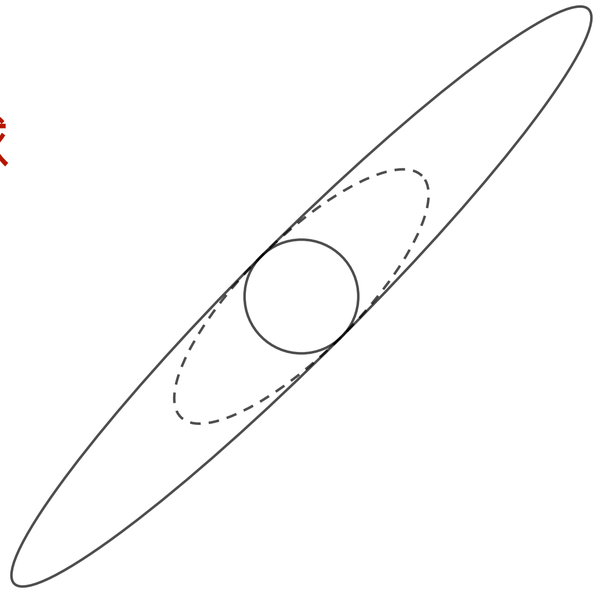
- 近似求（最大）principle component
  - Power iteration：线性时间

$\text{nnz}(\mathbf{X})$

近似求Top Principle Component: Power Iteration

算法设计思路

- 回忆：$\mathbf{A} = \mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$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$\mathbf{Q}\mathbf{e}_1$垂直，那么反复应用$\mathbf{A}$后：
  - 将在$\mathbf{Q}\mathbf{e}_1$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$\mathbf{Q}\mathbf{e}_1$的方向

如果看椭球，则会在$\mathbf{Q}\mathbf{e}_1$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$\mathbf{u}_i/\|\mathbf{u}_i\| \approx \mathbf{u}_{i-1}/\|\mathbf{u}_{i-1}\|$则停止并返回$\mathbf{u}_i/\|\mathbf{u}_i\|$

实现细节

复杂度nnz($\mathbf{X}$)

- 计算$\mathbf{A}u = \mathbf{X}^\top\mathbf{X}\mathbf{u}$时要注意顺序，先算$\mathbf{X}\mathbf{u}$而不是$\mathbf{X}^\top\mathbf{X}$
- $\mathbf{u}_i := \mathbf{A}^i\mathbf{u}_0$也不要每个$i$重新计算
- 利用递推式$\mathbf{u}_i = \mathbf{A}\mathbf{u}_{i-1} = \mathbf{X}^\top\mathbf{X}\mathbf{u}_{i-1}$

类似地，$\mathbf{A}^i\mathbf{u}_0$要利用$\mathbf{X}^\top\mathbf{X}u_{i-1}$从右往左计算，否则会从nnz($\mathbf{X}$)劣化成$nd^2$复杂性

作业十：利用Power Iteration求Top PC

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

Power Iteration的保证

结论：设$\mathbf{v}_1$是top PC，则对于任何$t \geq 1$，随机$\mathbf{u}_0$有常数概率使得

注意只能保证绝对值，
但对于找PC足够

$$|\langle \mathbf{u}_t, \mathbf{v}_1 \rangle| \geq 1 - O(\sqrt{d})\left(\frac{\lambda_2}{\lambda_1}\right)^t$$

λ_1/λ_2叫做spectral gap
实际数据经常gap是>1的常数

$|\langle \mathbf{u}_t, \mathbf{v}_1 \rangle|$衡量了算法输出与$\mathbf{v}_1$间的误差：
如果等于1就没误差，越接近1与$\mathbf{v}_1$方向越接近

假设结论成立，那么要达到$1-\epsilon$误差，只需要$t = O\left(\frac{\log(d/\epsilon)}{\log(\lambda_1/\lambda_2)}\right)$

算法分析

- 先考虑一下$\mathbf{A}^i$是什么
- $\mathbf{A}^i = (\mathbf{QDQ}^\top)^i = (\mathbf{QD}(\mathbf{Q}^\top\mathbf{Q})\mathbf{DQ}^\top)(\mathbf{QDQ}^\top)^{i-2} = \mathbf{QD}^2\mathbf{Q}^\top(\mathbf{QDQ}^\top)^{i-2}$
- 继续迭代下去，可以得到$\mathbf{A}^i = \mathbf{QD}^i\mathbf{Q}^\top$

$\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$

所以$\mathbf{A}^i$将所有eigenvalue做了i次幂

如果λ_1, λ_2有个gap，例如$\lambda_1 > 2\lambda_2$，
则gap会迅速变大

证明

随机的初始值$\mathbf{u}_0$在$\mathbf{v}_1$上已经有“足够”大的投影

orthonormal，可以看成一组基

设$\mathbf{Q}$的列是$\mathbf{v}_1, \ldots, \mathbf{v}_d$，这些也是$\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{D}\mathbf{Q}^\top$的eigenvectors

将随机向量$\mathbf{u}_0$表成$\mathbf{u}_0 = \sum_{i=1}^{d} a_i \mathbf{v}_i = \sum_{i=1}^{d} \langle \mathbf{u}_0, \mathbf{v}_i \rangle \mathbf{v}_i$

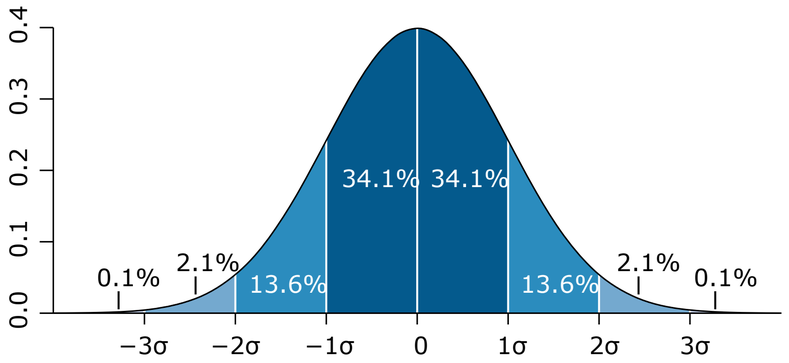
则常数概率有$|a_1| = |\langle \mathbf{u_0}, \mathbf{v_1} \rangle| \geq \Omega(1/\sqrt{d})$

为什么常数概率 $|a_1| = |\langle \mathbf{u_0}, \mathbf{v_1} \rangle| \geq \Omega(1/\sqrt{d})$?

$\mathbf{u}_0$是每维$\mathcal{N}(0,1)$再归一化得到的；先不归一化，考虑每维都是$\mathcal{N}(0,1)$的$\mathbf{u}'$

- 可得：$\langle \mathbf{u}', \mathbf{v}_1 \rangle$依然是一个$\mathcal{N}(0,1)$

根据正态分布，$|\langle \mathbf{u}', \mathbf{v}_1 \rangle| \geq 1$概率> 30%



再考虑对$\mathbf{u}'$的归一化：

- 归一化量等于 $\sqrt{\sum_{i=1}^{d} X_i^2}$

设$X_i \sim \mathcal{N}(0,1)$是独立高斯

开根号后在$\sqrt{d}$附近

结合大概率$|\langle \mathbf{u}', \mathbf{v}_1 \rangle| \geq 1$，得到$|a_1| \geq \Omega(1/\sqrt{d})$

- 这个高斯平方和就是一个chi-square distribution，大概率聚集在d附近

将$\langle \mathbf{A}^t\mathbf{u}_0/\|\mathbf{A}^t\mathbf{u}_0\|, \mathbf{v}_1\rangle$表示成 a_i, λ_i

$\mathbf{A}^t\mathbf{u}_0/\|\mathbf{A}^t\mathbf{u}_0\|$是算法输出的向量

事实：$\mathbf{A}^t\mathbf{u}_0 = \sum_{i=1}^{d} a_i \mathbf{Q}\mathbf{D}^t\mathbf{Q}^\top \mathbf{v}_i = \sum_{i=1}^{d} a_i \lambda_i^t \mathbf{v}_i$

因此$|\langle \mathbf{A}^t\mathbf{u}_0, \mathbf{v}_1\rangle| = |a_1\lambda_1^t|$；另外：

$\sum_i a_i^2 = 1$

$$\|\mathbf{A}^t\mathbf{u}_0\|^2 = \|\sum_{i=1}^{d} a_i\lambda_i^t\mathbf{v}_i\|^2 = \sum_{i=1}^{d}(a_i\lambda_i^t)^2 \le a_1^2\lambda_1^{2t} + (a_1^2 + \ldots + a_d^2)\lambda_2^{2t}$$

$$\le a_1^2\lambda_1^{2t} + \lambda_2^{2t}$$

完成证明

已有：常数概率有 $|a_1| = |\langle \mathbf{u_0}, \mathbf{v_1} \rangle| \geq \Omega(1/\sqrt{d})$

$$\frac{|\langle \mathbf{A}^t\mathbf{u}_0, \mathbf{v}_1 \rangle|}{\|\mathbf{A}^t\mathbf{u}_0\|} \geq \frac{|a_1\lambda_1^t|}{\sqrt{a_1^2\lambda_1^{2t} + \lambda_2^{2t}}} \geq \frac{|a_1\lambda_1^t|}{|a_1\lambda_1^t| + |\lambda_2^t|}$$

则：

$$= 1 - \frac{|\lambda_2^t|}{|a_1\lambda_1^t| + |\lambda_2^t|} \geq 1 - \frac{\lambda_2^t}{|a_1\lambda_1^t|}$$

$$\geq 1 - O(\sqrt{d} \cdot (\lambda_2/\lambda_1)^t)$$

如果要计算第二PC呢?

一个基本方法

- idea：先找top PC，“排除” top PC后在新矩阵运行power iteration找第二PC
- 算法：
  - 用power iteration找第一PC $\mathbf{v}_1$
  - 对每个数据点$\mathbf{x}_i$做$\mathbf{x}_i \mapsto \mathbf{x}_i - \langle \mathbf{x}_i, \mathbf{v}_1 \rangle \mathbf{v}_1$
  - 在新数据集上做power iteration找到的第一PC就是第二PC（的近似）
- 可以递归进行下去来找第三、第四…

可以理解成把第一PC的分量删除，此时整个数据都只有垂直于第一PC的部分了，但依然包含第二PC

计算第二PC：基本方法的问题及解决方案

- $\mathbf{x}_i \mapsto \mathbf{x}_i - \langle \mathbf{x}_i, \mathbf{v}_1 \rangle \mathbf{v}_1$后会导致不再稀疏
- 解决方案：power iteration中的随机$\vec{u}_0$从垂直于$\vec{\mathbf{v}}_1$的单位球面的部分均匀采样
- 如何操作?
  - 等价于对$\mathbf{u}_0$进行rejection sampling，当$\langle \mathbf{u}_0, \mathbf{v}_1 \rangle = 0$时返回
  - 约束$\langle \mathbf{u}_0, \mathbf{v}_1 \rangle = 0$是$\mathbf{u}_0$的$d$个坐标的方程；故$\mathbf{u}_0$最后一维被其他维线性表出
    - 因此：照常生成前$d-1$维，最后一维用方程解出来

简化：用± 1变量替代初始值的高斯

每维随机高斯的模长是随机的，但这里± 1向量的模长是确定的

- 可以把$\mathbf{u}_0$改成：每维是均匀± 1，最后再乘以归一化系数$1/\sqrt{d}$
- 数学结论：对一个每维均匀± 1向量$\mathbf{u}$和任意单位向量$\mathbf{v}$，有

$$\Pr[\langle \mathbf{u}, \mathbf{v} \rangle \geq 0.5] \geq \Omega(1)$$

- 所以常数概率，依然有$|\langle \mathbf{u}_0, \mathbf{v}_1 \rangle| \geq \Omega(1/\sqrt{d})$
- 之后的分析步骤都是一样的

扩展：如果$\lambda_1 = \lambda_2$呢?

思考：所有λ_i都想等呢?
是更难还是更简单?

- Power iteration依然会输出一个向量$\mathbf{u}$
- 但是性能保证不该继续看$\langle \mathbf{u}, \mathbf{v}_1 \rangle$，而是要$\mathbf{u}$在$\mathrm{span}(\mathbf{v}_1, \mathbf{v}_2)$上的投影尽量大
- 有类似的投影长度保证：

即找到一个“基本”在子空间中的向量

$\lambda_1 = \lambda_2$时principle component本来也不唯一，
任何一个落在$\mathrm{span}(\mathbf{v}_1, \mathbf{v}_2)$的向量都合法

投影长度 $\geq 1 - O(\sqrt{d})\left(\frac{\lambda_3}{\lambda_1}\right)^t$，也就是说可以继续看$\lambda_1$（或$\lambda_2$）与$\lambda_3$的gap

一般而言如果$\lambda_1 = \lambda_2 = \ldots = \lambda_i$，就可以有关于$\lambda_{i+1}/\lambda_1$的算法轮数保证

此时算法输出的是一个近似在$\mathbf{v}_1, \ldots, \mathbf{v}_i$ span里的向量

SVD

从rank说起

已知行与行之间互为一个倍数，请问如何补齐下面的矩阵?

$$\begin{bmatrix} 7 & ? & ? \\ ? & 8 & ? \\ ? & 12 & 6 \\ ? & ? & 2 \\ 21 & 6 & ? \end{bmatrix}$$

一般地，一个rank-1矩阵可以写成两个列向量$\mathbf{u}, \mathbf{v}$外积$\mathbf{u}\mathbf{v}^\top$的形式

$$\begin{bmatrix} 7 & 2 & 1 \\ 28 & 8 & 4 \\ 42 & 12 & 6 \\ 14 & 4 & 2 \\ 21 & 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 6 \\ 2 \\ 1 \end{bmatrix} [721] = \mathbf{u}\mathbf{v}^\top = \begin{bmatrix} -u_1\mathbf{v}^\top- \\ -u_2\mathbf{v}^\top- \\ \vdots \\ -u_m\mathbf{v}^\top- \end{bmatrix}$$

rank-k

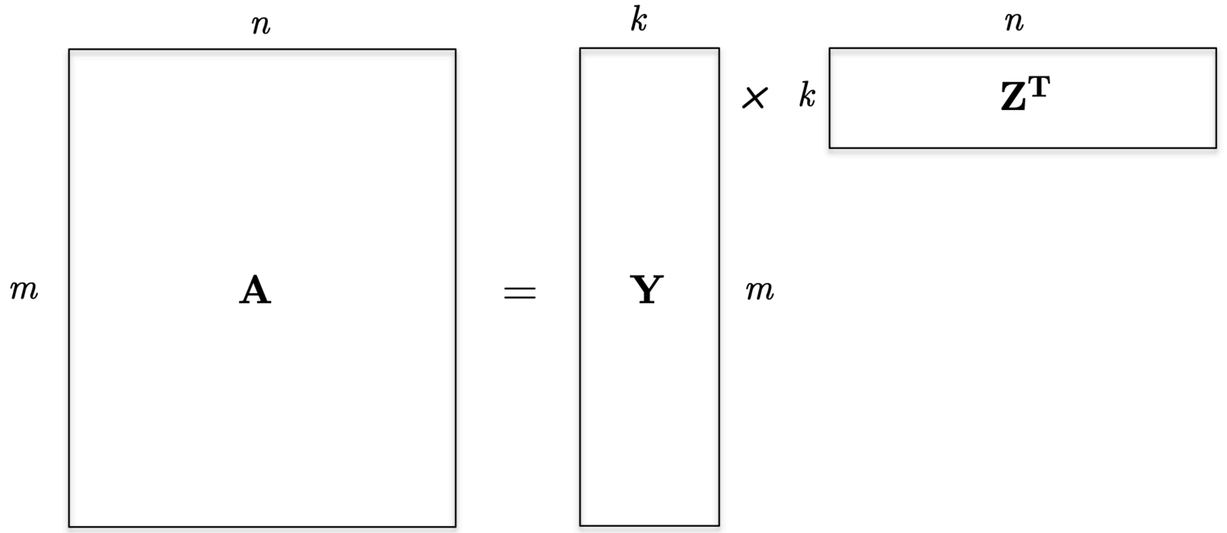
其他等价形式：例如最大的线性独立的行或者列的个数是k

- rank-k：可写成k个rank-1矩阵的和，并且不能写成任何$k-1$个rank-1矩阵和
  - 例如rank-2

$$\mathbf{A} = \mathbf{u}\mathbf{v}^\top + \mathbf{w}\mathbf{z}^\top = \begin{bmatrix} -u_1\mathbf{v}^\top + w_1\mathbf{z}^\top - \\ -u_2\mathbf{v}^\top + w_2\mathbf{z}^\top - \\ \vdots \\ -u_m\mathbf{v}^\top + w_m\mathbf{z}^\top - \end{bmatrix} = \begin{bmatrix} | & | \\ \mathbf{u} & \mathbf{w} \\ | & | \end{bmatrix} \begin{bmatrix} -\mathbf{v}^\top - \\ -\mathbf{z}^\top - \end{bmatrix}$$

rank-k矩阵的分解

根据刚才对于rank-2的讨论，一般而言，一个rank-k矩阵可以写成



A的n列是$\mathbf{Y}$的k列的线性组合，且A的m行是$\mathbf{Z}^\top$的k行的线性组合

SVD

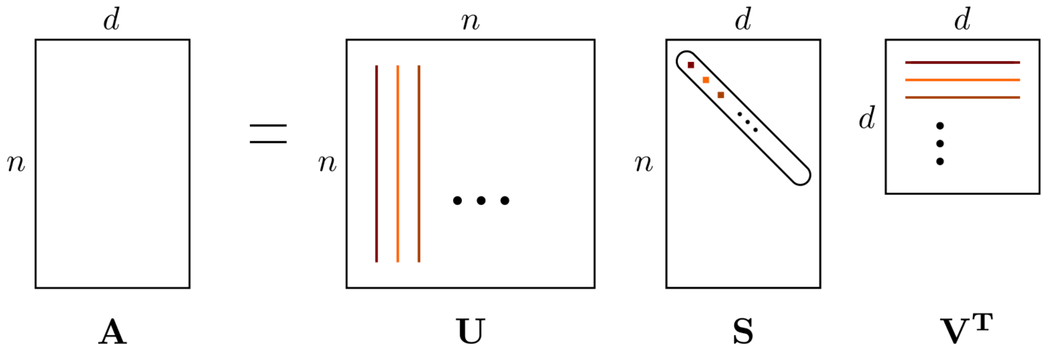
Singular Value Decomposition

U和V的每一列分别称作一个left或者right singular vector

- 任何矩阵$\mathbf{A}$可写成$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$

每个元素叫做singular value，非0的个数等于矩阵的rank

  - $\mathbf{U}, \mathbf{V}$都是orthogonal matrix；$\mathbf{S}$是对角阵，每个元素都非负，一般按照降序排列



与rank的关系：$\mathbf{A} = \sum_{i=1}^{\min\{n,d\}} s_i \mathbf{u}_i \mathbf{v}_i^\top$

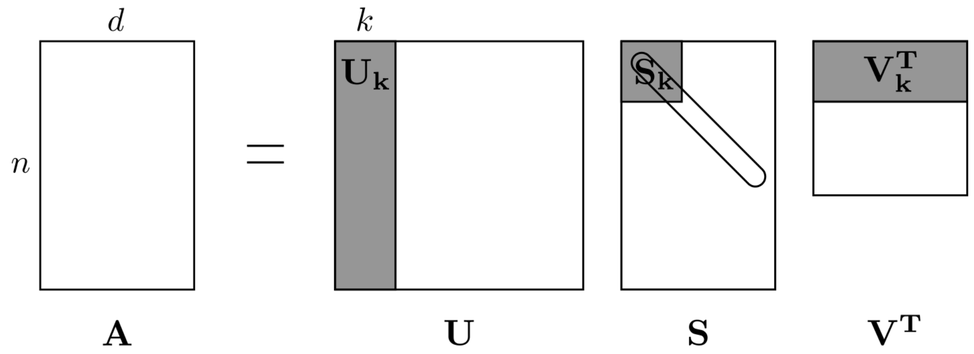
设rank是k，那么只有前k个s_i非零

每个$\mathbf{u}_i\mathbf{v}_i^\top$是一个rank-1矩阵，SVD就是一种显式将$\mathbf{A}$写成rank($\mathbf{A}$)个rank-1矩阵的方法

基于SVD的Low-rank Approximation

- 设$\mathbf{A}$的SVD是$\mathbf{USV}^\top$，设$\mathbf{U}_k, \mathbf{V}_k$分别为矩阵取前$k$列，$\mathbf{S}_k$为取前k行和列
- 考虑$\mathbf{A}_k := \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^\top$，那么$\mathbf{A}_k$的rank不超过k

注：$\mathbf{A}$和$\mathbf{A}_k$的维度是一样的，只是rank不同



$\mathbf{A}_k$的高效表示

等价写法：$\mathbf{A}_k = \sum_{i=1}^{k} s_i \cdot \mathbf{u}_i \mathbf{v}_i^\top$

- 即rank-1求和的表达中只取singular value最大的k个rank-1矩阵

- 因此存储$\mathbf{A}_k$的代价非常小，是$k(n+d)$，而直接存储就是nd
- 而且与$\mathbf{A}_k$有关的矩阵计算也可以是与k有关的快速计算，例如与向量相乘$\mathbf{A}_k\mathbf{x}$

时间复杂度是多少?

$\mathbf{A}_k$是最优的Low-rank Approximation

结论：对任何$n \times d$矩阵$\mathbf{A}$，任何$k \geq 1$，任何rank-k的$n \times d$矩阵$\mathbf{B}$，有

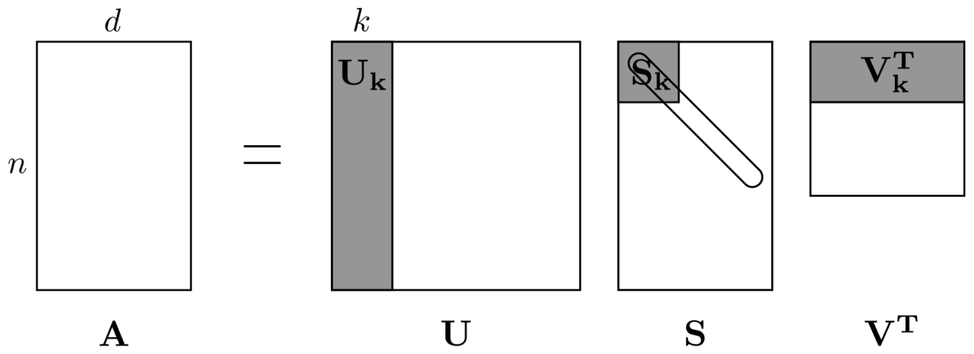
$$\|\mathbf{A} - \mathbf{A}_k\|_F \leq \|\mathbf{A} - \mathbf{B}\|_F$$

也就是说在F范数的误差衡量下，$\mathbf{A}_k$具有最小的误差

其中对于任意矩阵$\mathbf{M}$，$\|\mathbf{M}\|_F := \sqrt{\sum_{i,j} M_{ij}^2}$

Frobenius norm

最优性的一些Intuition



在$\mathbf{A} = \sum_{i=1} s_i \cdot \mathbf{u}_i \mathbf{v}_i^\top$表示中，$\mathbf{u}_i \mathbf{v}_i^\top$的“大小”是一样的

因为都是orthonormal的

但是singular value，s_i，决定了每项$\mathbf{u}_i \mathbf{v}_i^\top$的显著性

$\mathbf{A}_k = \sum_{i=1}^{k} s_i \cdot \mathbf{u}_i \mathbf{v}_i^\top$取最显著的k项，直觉上是对$\mathbf{A}$某种意义上最好的近似

Low-rank Approximation的典型应用

- 压缩/降维：nd到$k(n+d)$
- 去噪声：如果输入$\mathbf{A}$是某个low-rank数据源的带噪声采样
  - 那么选择合适的rank的low-rank approximation即可去噪声、精度损失小
- 矩阵还原
  - 很多矩阵有不少缺失值，尤其是例如电商网站用户-商品评分矩阵
  - 先将缺失项补0或者列均值，然后进行low-rank approximation

若原矩阵确实有low-rank结构且没有缺失太过多的信息，那么这是一种不错的还原方法

SVD与PCA的关系

SVD可以解PCA

一句话：SVD可以规约到PCA（即可以用SVD解PCA）

设$\mathbf{A} = \mathbf{X}^\top\mathbf{X}$，然后设$\mathbf{X}$的SVD为$\mathbf{X} = \mathbf{USV}^\top$，则

$$\mathbf{A} = \mathbf{X}^\top\mathbf{X} = (\mathbf{USV}^\top)^\top(\mathbf{USV}^\top) = \mathbf{VS}^\top\mathbf{U}^\top\mathbf{USV}^\top = \mathbf{VU}^2\mathbf{V}^\top$$

也就是写成了$\mathbf{QDQ}^\top$的形式，因此$\mathbf{V}$的列就是$\mathbf{X}$的principle components

PCA只需要用$\mathbf{V}$，不需要$\mathbf{U}$

经常可以听到人说“PCA就是SVD”

解SVD：SVD求解有很成熟的solver，不建议自己写，最好直接调用

SVD与PCA的关系

Power iteration可以（部分）解SVD

- 另一个方向：PCA解SVD?
- 注意到设某个一般$\mathbf{X}$的SVD是$\mathbf{X} = \mathbf{USV}^\top$则我们已经得出

$$\mathbf{A} = \mathbf{X}^\top\mathbf{X} = (\mathbf{USV}^\top)^\top(\mathbf{USV}^\top) = \mathbf{VS}^\top\mathbf{U}^\top\mathbf{USV}^\top = \mathbf{VU}^2\mathbf{V}^\top$$

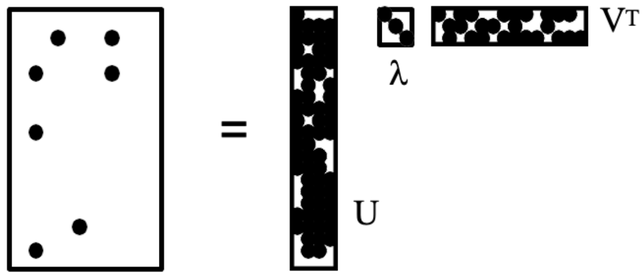
- 那么本来是用来解PCA的power iteration，也可以用来近似求$\mathbf{V}$的第一列
  - 也就是可以用来解一般矩阵$\mathbf{X}$的right singular vectors

降维科研进展选讲

* SVD的局限性

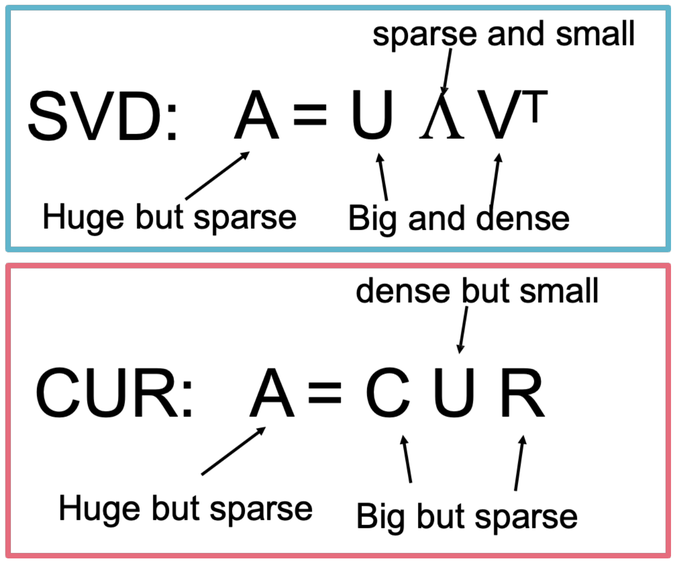
- SVD中$\mathbf{USV}^{\top}$的$\mathbf{U}$和$\mathbf{V}$都是dense的
  - 即使输入是sparse的，他们也还是dense的！
  - 另外它们不是原数据的行列，可解释性一般

U和V只是与数据的行/列具有某种隐性关系



另一种Low-rank Approximation

- CUR decomposition
- 给定目标的rank k，把矩阵$\mathbf{A}$写成$\mathbf{A} = \mathbf{CUR}$
  - $\mathbf{U}$是dense的，但是rank是k
  - $\mathbf{C}$和$\mathbf{R}$分别是$\mathbf{A}$的某些列和行
- 目标：让$\|\mathbf{A} - \mathbf{CUR}\|_F \approx \|\mathbf{A} - \mathbf{A}_k\|_F$



构造CUR的一种简单方法

- 按照行/列的ℓ_2范数进行重要性采样
- 会产生$\epsilon\|\mathbf{A}\|_F$的additive误差

[Frieze, Kannan, Vempala, JACM 04]

Sampling columns (similarly for rows):

Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$ [column distribution]
2. $P(x) = \sum_i \mathbf{A}(i,x)^2 / \sum_{i,j} \mathbf{A}(i,j)^2$
3. for $i = 1 : c$ [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(x)$
5. Compute $\mathbf{C}_d(:,i) = \mathbf{A}(:,j) / \sqrt{cP(j)}$

采样次数是poly(k/ϵ)

CUR的State of the art

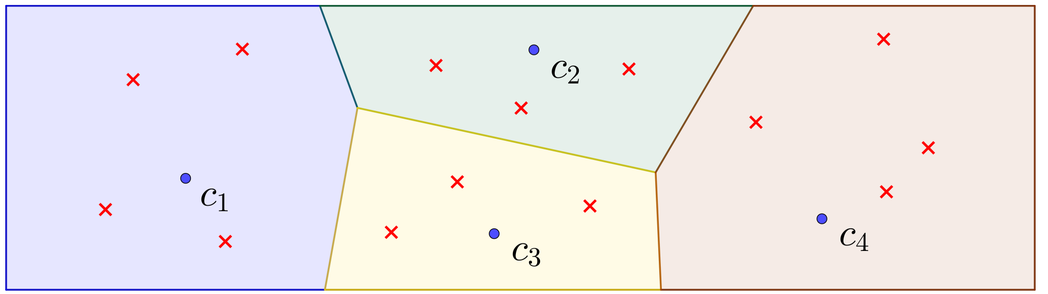
- [Boutsidis, Woodruff, STOC 14]
- $\mathbf{C}$和$\mathbf{R}$大小都是$O(k/\epsilon)$，$\mathbf{rank}(\mathbf{U}) = k$并且

$$\|\mathbf{A} - \mathbf{CUR}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F$$

运行时间：$\mathrm{nnz}(\mathbf{A}) \cdot \mathrm{poly}\log n + \mathrm{poly}(k/\epsilon)$

* 一个重要数据分析问题：聚类

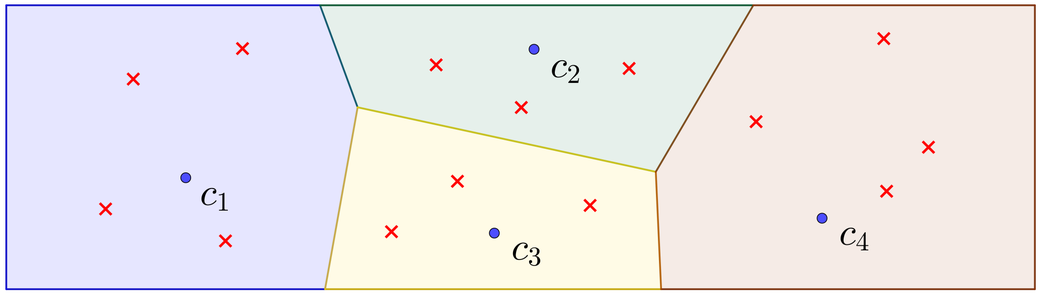
- 我们讲过的linear regression和PCA都是最fundamental的数据分析方法
- 还有一个同等重要的：聚类
- 我们聚焦于k-means clustering
- 输入数据$P = (x_1, \ldots, x_n) \subset \mathbb{R}^d$
- 目标是找聚类中心$C = (c_1, \ldots, c_k) \subset \mathbb{R}^d$使得目标函数最小化：



$$\text{cost}(P, C) := \sum_{x_i \in P} \min_{j=1}^{k} \|x_i - c_j\|^2$$

数据点到聚类中心最近距离的平方和

等价表述



$P_1, \ldots, P_k$对应k个cluster

- 找到一个数据点集P的k-划分$\mathscr{C} = \{P_1, \ldots, P_k\}$，使得目标函数最小化

求该cluster数据点到中心距离的平方和

$$\mathrm{cost}(P, \mathscr{C}) := \sum_{i=1}^{k} \min_{c_i \in \mathbb{R}^d} \sum_{x \in P_i} \|x - c_i\|^2$$

枚举cluster编号

寻找第i个cluster中心c_i

为什么是等价表述：因为最优聚类中心对应的最优划分一定是按照最近邻规则的

$$\mathrm{cost}(P, C) := \sum_{x_i \in P} \min_{j=1}^{k} \|x_i - c_j\|^2$$

也就是最优的聚类中心和最优划分在两个函数下是一样的

JL类型的聚类降维

- JL：设target dimension $m \approx \frac{\log k}{\epsilon^2}$，远好于我们学的$\frac{\log n}{\epsilon^2}$!
- 结论：将每个数据点x_i用JL变换到m维，记新数据集$P_{\text{low}} \subset \mathbb{R}^m$

[Makarychev-Makarychev-Razenshteyn, STOC 19]

  - 对任何P的k-划分$\mathscr{C}$，有

P_{low}和P有一一映射，$\mathscr{C}$在P_{low}上有对应意义

$$\text{cost}(P_{\text{low}}, \mathscr{C}) \in (1 \pm \epsilon) \cdot \text{cost}(P, \mathscr{C})$$

- 因此：只需要在只有$O(\log k)$维的P_{low}上求解k-means

SVD类型的聚类降维

- 把n个d维数据点$x_1, \ldots, x_n$安排成一个$n \times d$矩阵$\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} -x_1- \\ \vdots \\ -x_n- \end{bmatrix}$$

被证明不可改进

[Cohen et al., STOC 14]

- 结论：对$\mathbf{X}$做SVD，取$m = \lceil k/\epsilon \rceil$为rank的low-rank SVD近似，得到$\mathbf{X}_m$
  - 则$\mathbf{X}_m$上的最优聚类与$\mathbf{X}$上的最优聚类只差$(1 \pm \epsilon)$倍