

非监督学习 —— 降维

卿来云

大纲

■ 简介

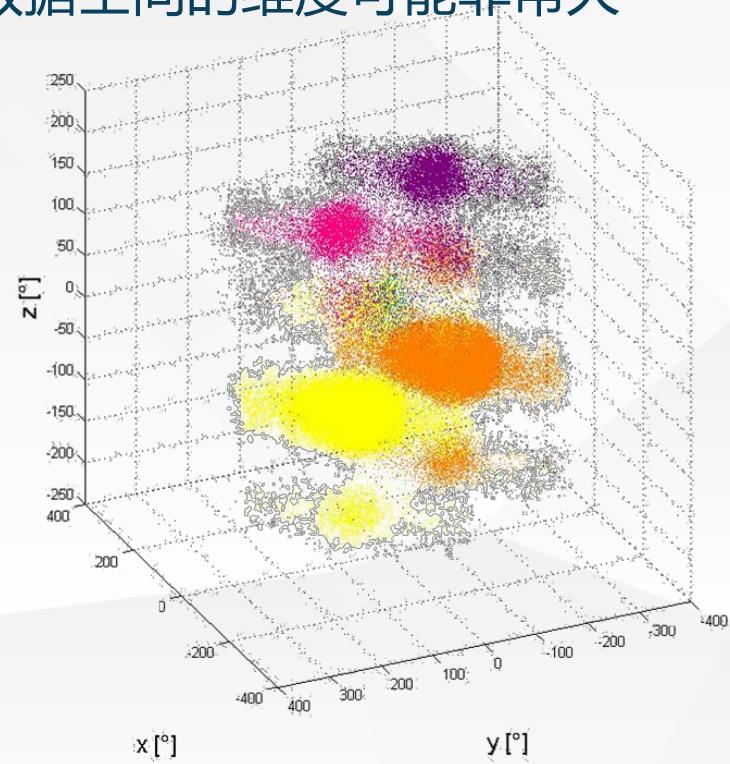
■ 降维算法

- 维度选择
- 维度抽取
 - 1. 预备知识
 - 2. 线性模型
 - 3. 非线性模型

➤ 关于维度——数据空间

■ 维度 = 特征

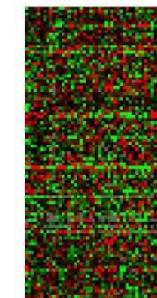
■ 数据空间的维度可能非常大



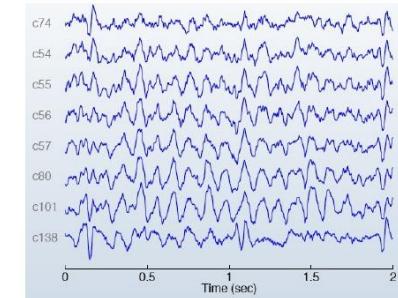
face images

According to media reports, a pair of hackers said on Saturday that the Firefox Web browser, commonly perceived as the safer and more customizable alternative to market leader Internet Explorer, is critically flawed. A presentation on the flaw was shown during the ToorCon hacker conference in San Diego.

documents



gene expression data



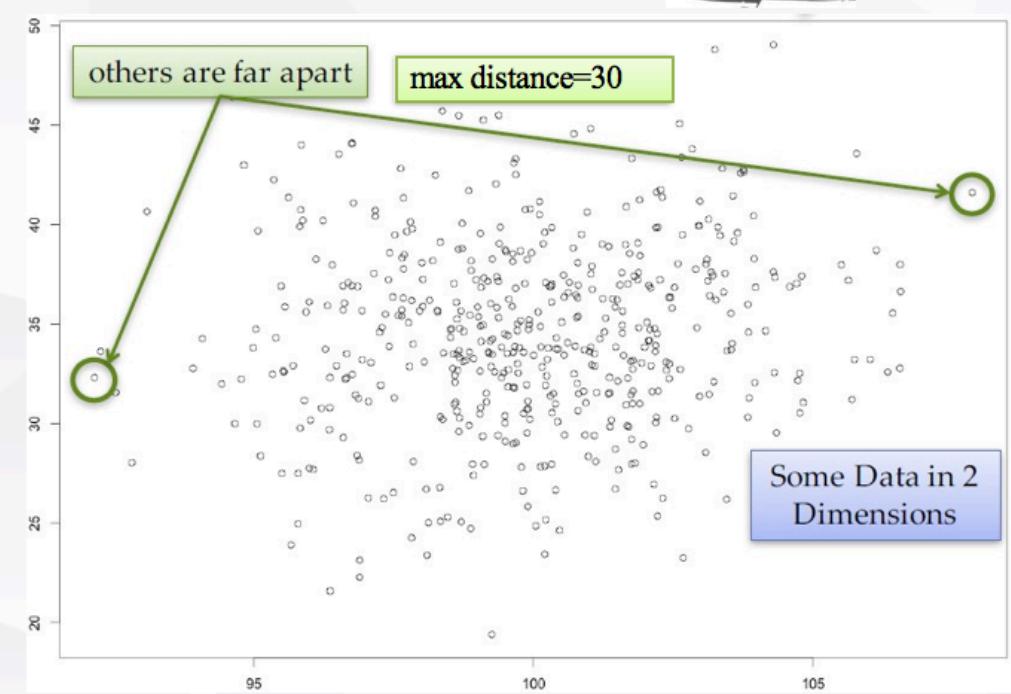
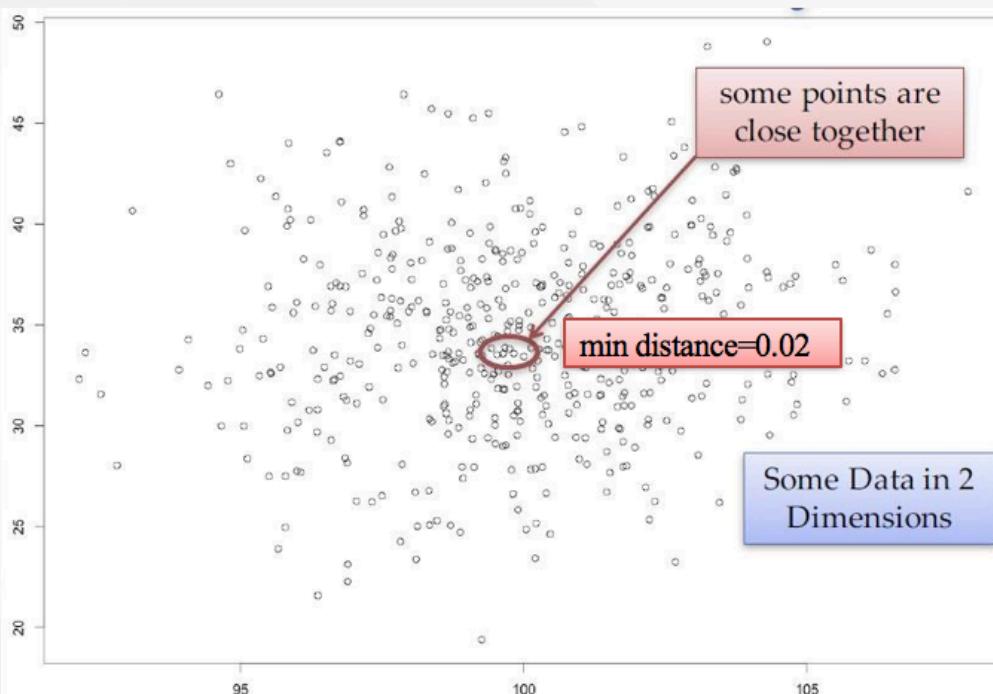
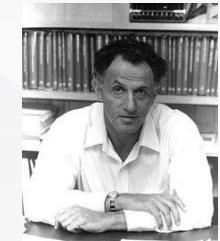
MEG readings

非常多高维的数据

维数灾难

- 随着空间维度的增长，数据点越来越分散，以至于距离和密度的概念变得越来越模糊

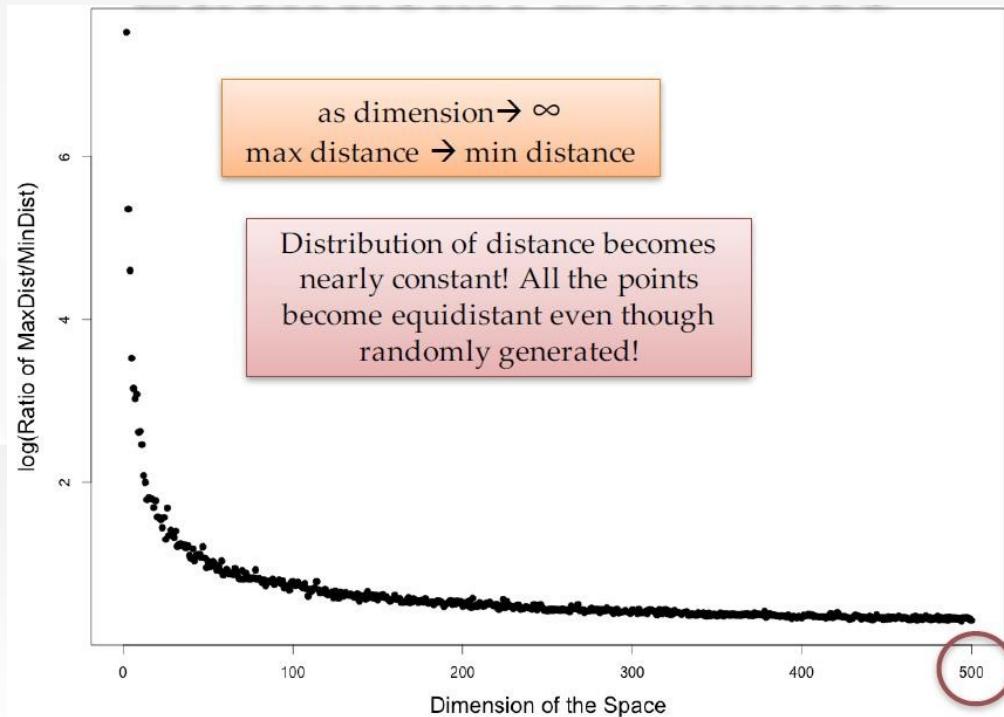
Richard Ernest Bellman



$$\text{max/min} = 30/0.02 = 1500. \text{ 最大距离是最小距离的1500倍.}$$

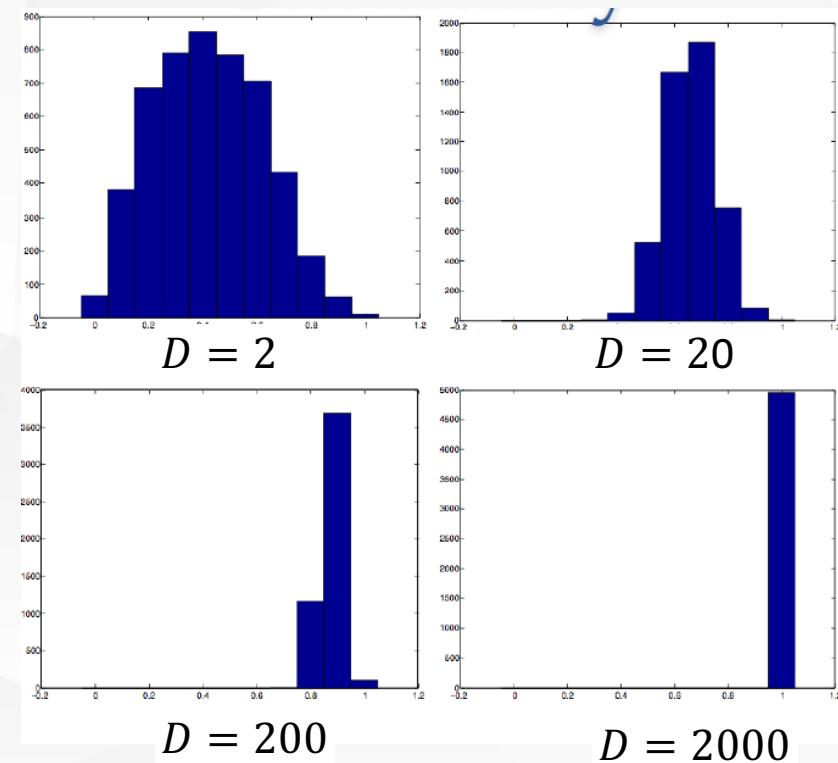
维数灾难

- 生成500个数据点从3维/4维/.../500维空间
- 计算最大距离与最小距离的比值
- 观测该比值随着维度增长的变化



维数灾难

- 没有一种距离函数或相似性函数能在避免高维带来的问题
- 例如：余弦相似度的分布
- 计算在任意点对的余弦相似度，除以最大值进行归一化



➤ 什么时候维数灾难会产生影响

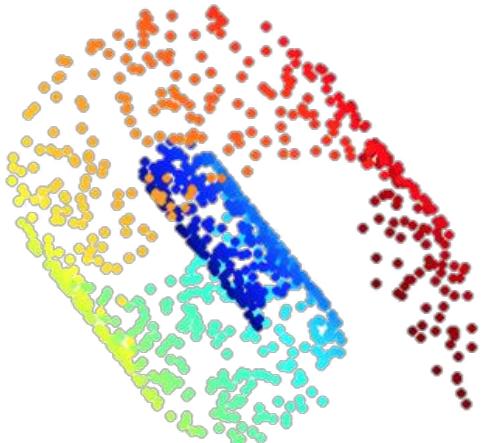
- 当我们使用的算法需要用到距离和相似性计算时
 - 尤其对于聚类和 K 近邻算法
- 由于共线性引起的问题，想得到一个简单模型
 - 冗余特征
- 算法的计算和存储复杂度成为难题

针对此我们能做什么？

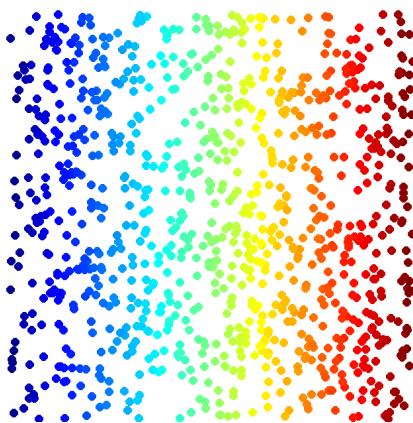
降维/嵌入

降维

- 降维或嵌入是指将原始的高维数据映射到低维空间。
- 实质的想法：高度冗余的数据通常是可以被压缩的，即高维复杂的数据其内在的维度可能比较小，或者和任务相关的维度较小。



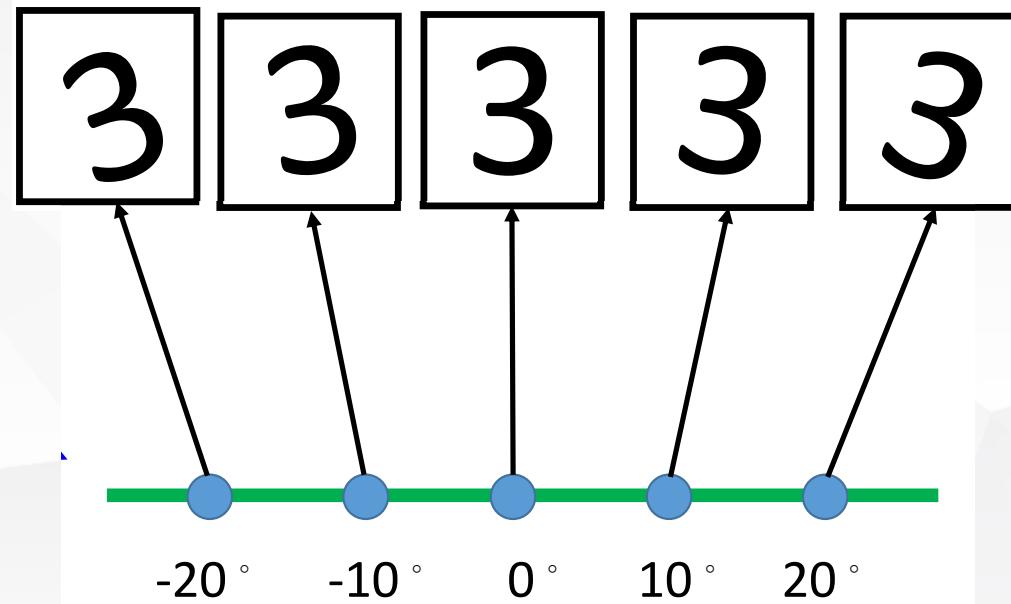
3D数据



本质维度为2D

➤ 示例1: 手写字符识别

- 特征表示: 一个高维向量 (e.g., $28 \times 28 = 784$) 每个维度表示像素的亮度值
- 本质结构参数: 旋转方向 (1维)



➤ 示例2: 文本文档分析

- 表示: 高维向量 (例如, 10K) , 每个维度对应词表里的一个词



Term	D1	D2
game	1	0
decision	0	0
theory	2	0
probability	0	3
analysis	0	2
...		

- 潜在的结构参数: 主题/话题

降维方法概览

■ 考虑数据集 \mathbf{X} 包含 D 维空间中的 N 个点

■ 数据集可以被看作 $D \times N$ 矩阵

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & \dots & x_{1,N} \\ \vdots & \ddots & \vdots \\ x_{D,1} & \dots & x_{D,N} \end{pmatrix}$$

■ 降维方法

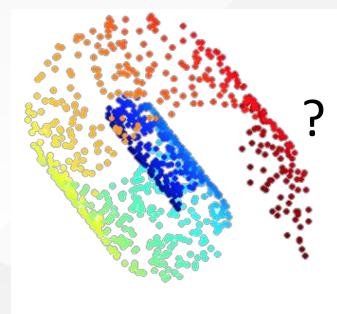
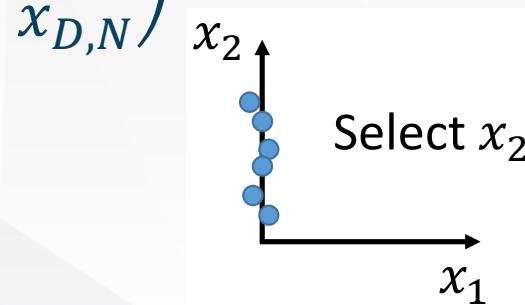
维度选择：选择已有维度的一个子集

维度抽取：通过组合已有的维度构建新的维度

■ 两种法案都是将原始空间 \mathbb{R}^D 中的数据点 \mathbf{x} 映射到新空间 $\mathbb{R}^{D'}$ 中的点 \mathbf{z}

■ 映射： $f: \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$

■ 通常： $D' \ll D$



大纲

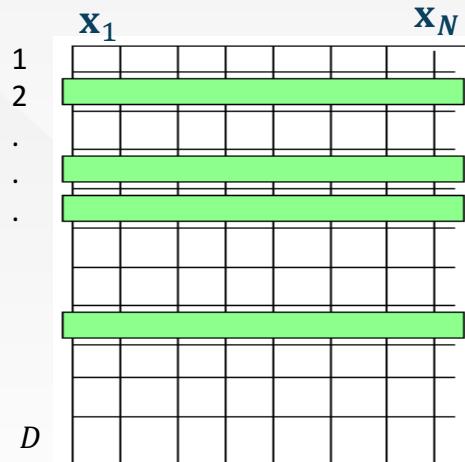
■ 简介

■ 降维算法

- 维度选择
- 维度抽取
 - 1. 预备知识
 - 2. 线性模型
 - 3. 非线性模型

➤ 维度选择

■ 采样: 从 D 维中随机选择 D' 维



$$\mathbf{X} \in \mathbb{R}^{D \times N} \Rightarrow \mathbf{Z} \in \mathbb{R}^{D' \times N} \quad \left(\mathbf{x}_1^T \mathbf{x}_2 = \sum_{d=1}^D x_{1,d} x_{2,d} \right) \approx \left(\mathbf{z}_1^T \mathbf{z}_2 = \sum_{d=1}^{D'} z_{1,d} z_{2,d} \right) \times \frac{D}{D'}$$

Johnson-Lindestrauss (JL) embedding lemma

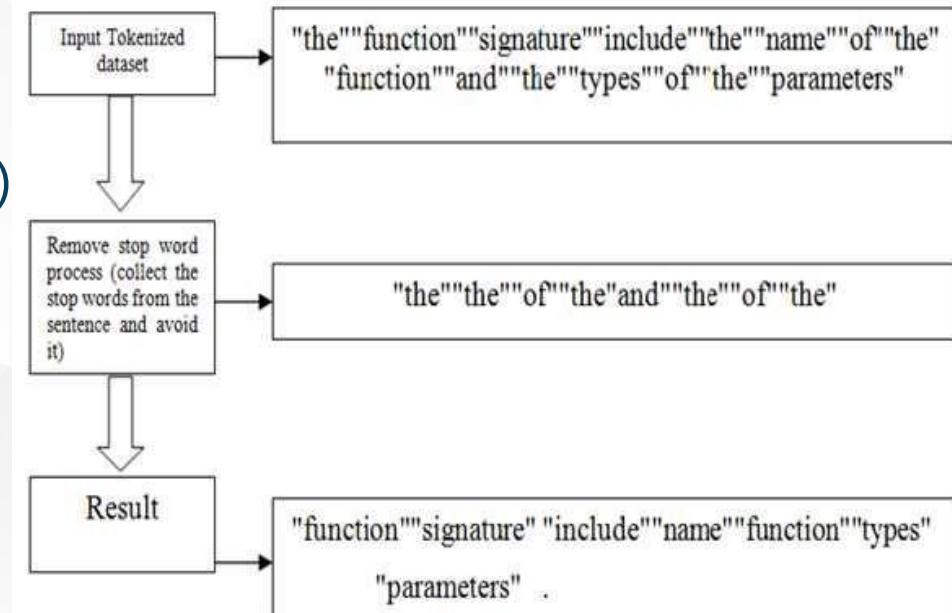
一个 D 维空间中的 N 个点可以近似等距地嵌入到一个 $D' \approx O(\log N)$ 维的空间。

- **优点**: 简单, 流行, 具有较好泛化性能 (不止近似距离)
- **缺点**: 没有精度保证。差的例子上错误很大 (重尾分布), 稀疏数据上大多数是0.

➤ 维度选择

■ 手工: 手工移除特征

- 冗余的 (multicollinearity/VIFs)
- 不相关 (文本挖掘中的停用词)
- 质量差的特征 (特征的缺失比例超过50%)
- 方差过小的特征



sklearn.feature_selection

https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection

➤ 过滤式选择

■ 过滤式维度选择通过设计一个相关统计量来度量特征的重要性。

■ 统计量

- 相关系数：单个特征与标签 y 之间的相关系数（回归问题）
- 互信息：单个特征和分类标签 y 的互信息（ y 为离散值）
- χ^2 统计量：单个特征和分类标签 y 之间的相关性
- ...

■ 过滤式维度选择根据单个特征与目标之间统计分数选择特征，速度快，但缺点是没有考虑到特征之间的关联作用。

➤ 例：特征选择：文本分类

■ 文本分类中的过滤型特征选择（英文中大约有 10^5 词）

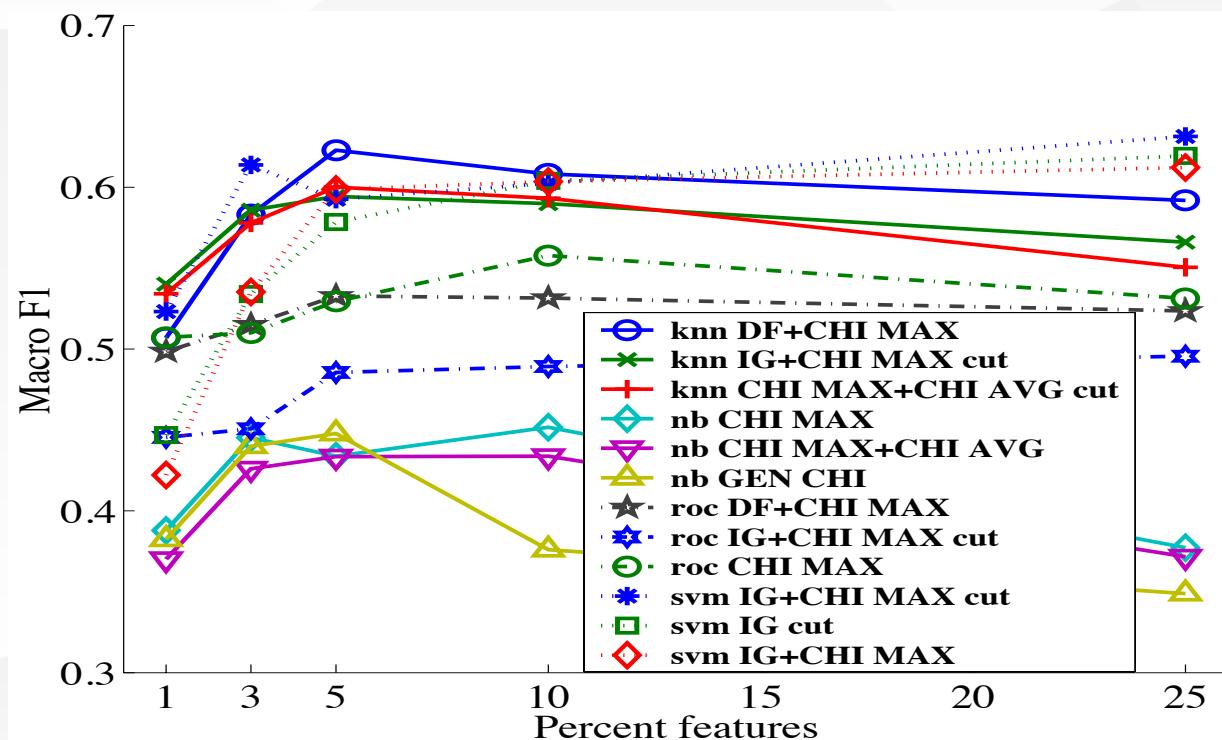


Figure 2: Top 3 feature selection methods for Reuters-21578 (Macro F1)

Monica Rogati, Yiming Yang, High-Performing Feature Selection for Text Classification, CIKM2012

➤ 包裹式选择

■用最终要用的学习器的性能评价特征的重要性。

■搜索有用的特征子集

- 前向
 - 从零个特征开始
 - 一遍式(one pass)或者迭代式(iterative)地选择
- 后向（更常用，递归特征删除）
 - 用所有特征训练一个模型，得到特征重要性
 - 根据特征重要性，每次删除最不重要的一些特征

■是否删除/增加特征，需要进行模型性能监控（如校验集上的性能）。

➤ 嵌入式选择

■ 嵌入式维度选择与模型训练一起完成。

■ 基于L1正则的特征选择

- L1正则得到的系数可能是稀疏的，选择非0系数即可实现特征选择
Lasso、Logistic回归（L1正则）、SVM（L1正则）

■ 基于树模型的特征选择

- 在基于树的模型（CART、随机森林、GBDT等）中，树中从根节点到叶子结点的路径上的特征，为模型选择的特征，不在树中出现的特征没有被模型选中。

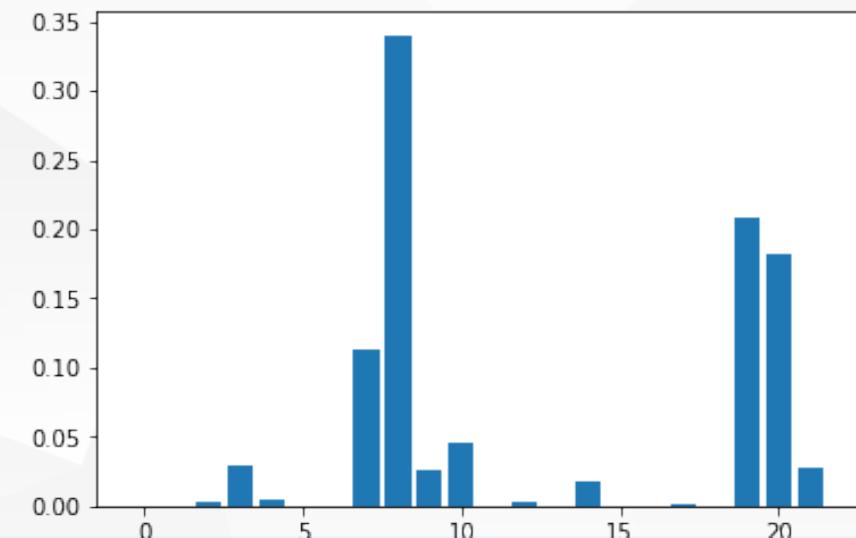
➤ 例：嵌入式维度选择

例：商品销量与广告费用（线性回归）

特征	最小二乘线性 回归系数	岭回归系数	Lasso系数
TV	3.983944	3.981524	3.921642
radio	2.860230	2.858304	2.806374
newspaper	0.038194	0.038925	0.000000
截距项	13.969091	13.969282	13.972528

决策树的特征重要性

例：蘑菇毒性判断：



大纲

- 简介
- 降维算法
 - 维度选择
 - 维度抽取
- 1. 预备知识
- 2. 线性模型
- 3. 非线性模型

➤ 矩阵

■ 矩阵是一个由 M 行 (row) N 列 (column) 元素排列成的矩形阵列。矩阵的元素可以是数字、符号或表达式。

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{1,1} & a_{2,2} & \dots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \dots & a_{M,N} \end{bmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{1,1} & a_{2,2} & \dots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \dots & a_{M,N} \end{pmatrix} = (a_{i,j}) \in \mathbb{R}^{M \times N}$$

■ 给定矩阵 $\mathbf{A}_{M \times N}$ ，矩阵 \mathbf{A} 的秩是其最大线性无关行(或列)的数目

■ 代表了由行或列向量张成的维度空间

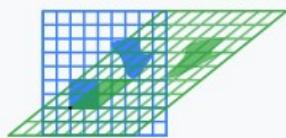
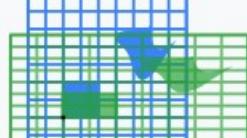
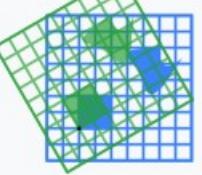
- Row Rank = Column Rank
- $rank(\mathbf{A}) \leq \min(M, N)$
- $rank(\mathbf{AB}) \leq \min(rank(\mathbf{A}), rank(\mathbf{B}))$

$$\begin{pmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{pmatrix} \quad \begin{bmatrix} 1 & 1 & 0 & 2 \\ -1 & -1 & 0 & -2 \end{bmatrix} \quad \begin{matrix} rank=2 \\ rank=1 \end{matrix}$$

■ 矩阵的迹：方阵 \mathbf{A} 的 $tr(\mathbf{A})$ 是其对角线上元素的和

➤ 矩阵乘法

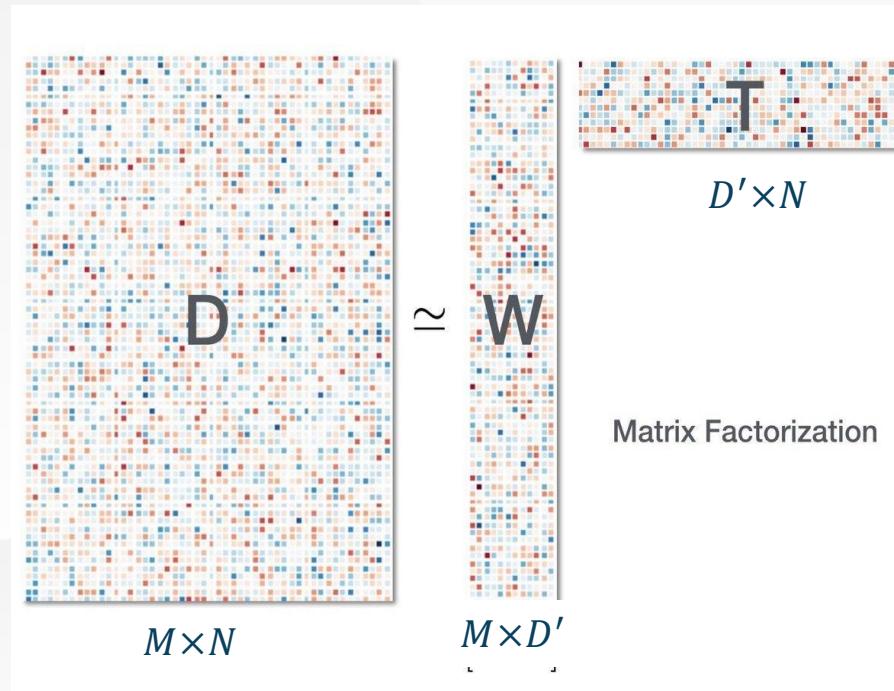
- 矩阵和矩阵乘法本质上是在做线性变换。
- 一个 $M \times N$ 实值矩阵 A 对应一个线性变换 $\mathbb{R}^N \rightarrow \mathbb{R}^M$ ，将向量 $x \in \mathbb{R}^N$ 映射到结果向量 $Ax \in \mathbb{R}^M$ 。

Horizontal shear with $m=1.25$.	Reflection through the vertical axis	Squeeze mapping with $r=3/2$	Scaling by a factor of 3/2	Rotation by $\pi/6^R = 30^\circ$
$\begin{bmatrix} 1 & 1.25 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 3/2 & 0 \\ 0 & 2/3 \end{bmatrix}$	$\begin{bmatrix} 3/2 & 0 \\ 0 & 3/2 \end{bmatrix}$	$\begin{bmatrix} \cos(\pi/6^R) & -\sin(\pi/6^R) \\ \sin(\pi/6^R) & \cos(\pi/6^R) \end{bmatrix}$
				

推荐阅读：理解矩阵（孟岩）：<https://blog.csdn.net/myan/article/details/647511>

➤ 矩阵分解

- 矩阵分解是将一个矩阵分解为几个矩阵的乘法
- 高维矩阵的**低秩近似**： D' 为隐含向量的维数



➤ 特征分解

- 输入：方阵 $A_{M \times M}$
- 特征向量和特征值： $Av = \lambda v$
 - 其中 v 是矩阵 A 的特征向量， λ 是对应的特征值， $V^T V = I$
- 特征分解（对角化）： $A = V \Lambda V^{-1}$
 - V 是矩阵的特征向量， Λ 是由特征值组成的对角矩阵。

$$A = V \Lambda V^{-1}$$

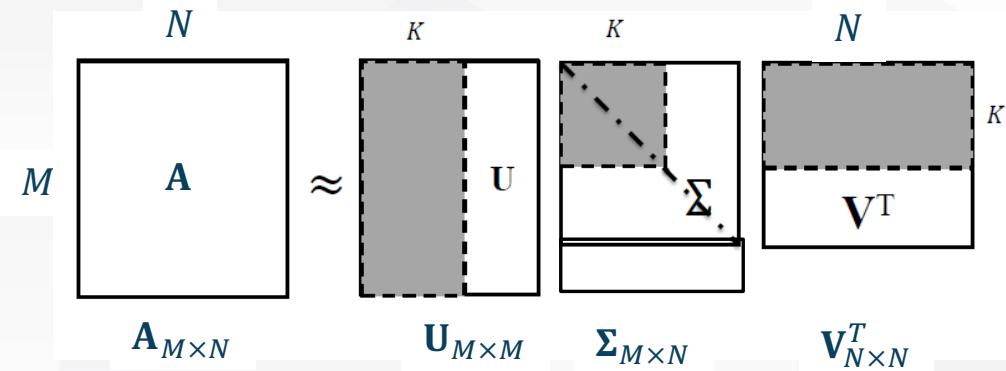
The diagram illustrates the matrix decomposition $A = V \Lambda V^{-1}$. It features four boxes labeled A, V, Λ, and V⁻¹. Box A contains a 3x3 grid of lines. Box V contains three vertical vectors labeled v₁, v₂, and v₃, with a green bracket below them labeled "Eigen vectors of A". Box Λ is a 3x3 diagonal matrix with entries λ₁, λ₂, and λ₃ on the diagonal. Box V⁻¹ contains the same three vectors v₁, v₂, and v₃, with a green bracket below them labeled "Eigen vectors of A". Brackets connect the corresponding columns of V and V⁻¹. The equation A = V * Λ * V⁻¹ is positioned between the boxes.

➤ 奇异值分解

- 输入：矩阵 $\mathbf{A}_{M \times N}$
- SVD : $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$

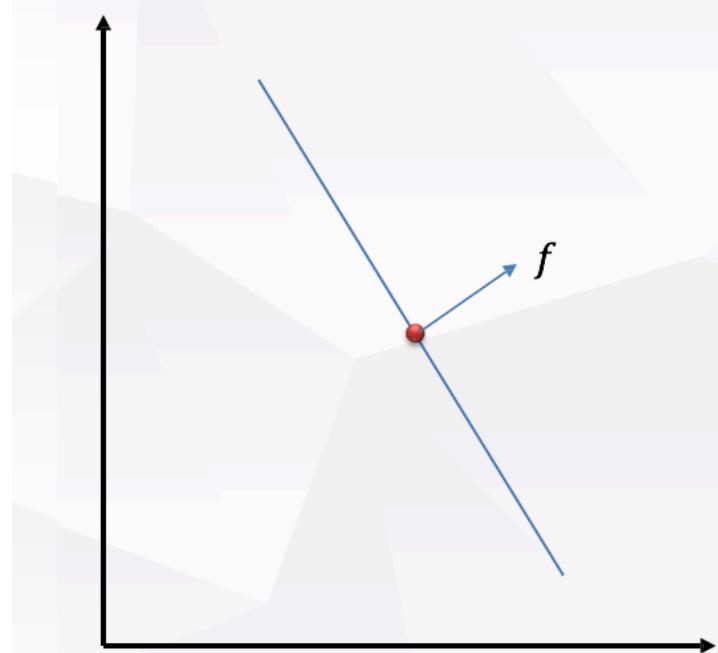
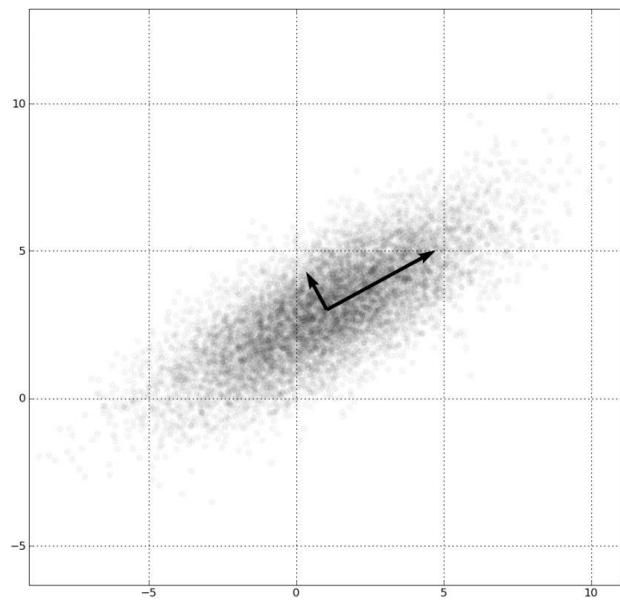
$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_R \end{bmatrix} \text{: 奇异值 , } R = \min(M, N)$$

\mathbf{u}_k 、 \mathbf{v}_k^T : 奇异值 σ_k 对应的奇异向量
 $\mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{V}^T\mathbf{V} = \mathbf{I}$: \mathbf{U} 和 \mathbf{V} 是正交矩阵



➤ 特征值或奇异值的物理意义

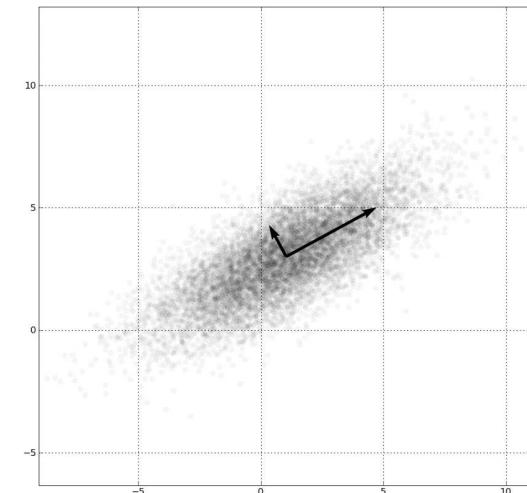
- 从统计的角度：方差
- 从物理的角度：能量



➤ 奇异值向量的含义

- $\mathbf{U}(\mathbf{V})$ 的每个列或行代表一个方向
- 列(行)之间相互正交
- 如果我们把 Σ 中的奇异值降序排列, 并且 $\mathbf{U}(\mathbf{V})$ 中 $\mathbf{u}_i(\mathbf{v}_i^T)$ 也相应的调整
 - \mathbf{u}_1 代表最大能量的方向
 - \mathbf{u}_2 代表和 \mathbf{u}_1 正交的能量最大的方向
 - \mathbf{u}_3 代表与 \mathbf{u}_1 和 \mathbf{u}_2 正交的能量最大的方向

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$



大纲

- 简介

- 降维算法

- 维度选择

- 维度抽取

1. 预备知识

2. 线性模型

3. 非线性模型

最简单的降维：随机投影择

- 随机投影：随机生成一个 $D' \times D$ 的矩阵 W ，将 \mathbb{R}^D 空间向量 x 投影为一致随机的 D' 维空间中的向量

(Johnson-Lindestrauss (JL) embedding lemma)

一个 D 维空间中的 N 个点可以近似等距地嵌入到一个 $D' \approx O(\log N)$ 维的空间。

$$z = \sqrt{\frac{D}{D'}} Wx$$

乘以系数 $\sqrt{\frac{D}{D'}}$ 是为了保证 $\mathbb{E} \left(\sqrt{\frac{D}{D'}} Wx \right)^2 = \|x\|^2$

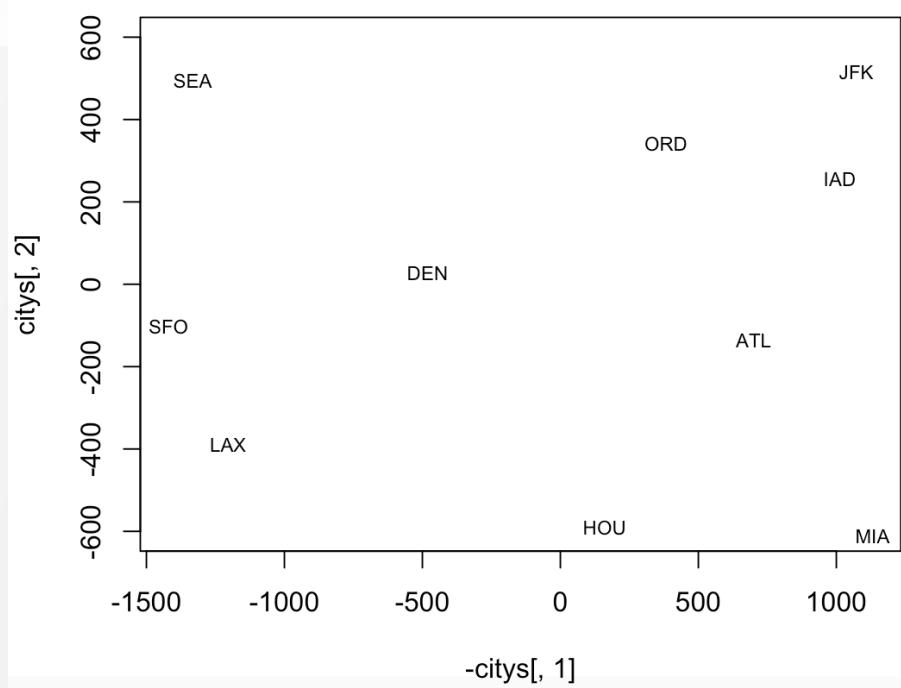
- Pros: 简单, 流行, 具有较好泛化性能 (不止近似距离)
- Cons: 没有精度保证

➤ 多维缩放(Multiple Dimensional Scaling, MDS)

■ 问题形式化:

- 给定空间中任意两个点的距离，点的精确坐标和维度是未知的。
 - 我们希望将这些点嵌入到一个低维的空间中，使得新的空间中点对之间的距离和原始空间中的距离尽可能接近。
- 显然，投影到低维空间的形式不是唯一的，原因是低维空间的点经过平移(translation),旋转(rotation)和镜像 (reflection)后，点对之间的距离不变。

➤ 绘制地图（城市嵌入）



在平面坐标上据此标出这10个城市之间的相对位置，使之尽可能接近表中的距离数据



➤ MDS的形式化

- 令矩阵 $\mathbf{Z} \in \mathbb{R}^{D' \times N}$ 表示所有样本点在 D' 维（低维）空间中的表示
基本思想： D' 空间的欧式距离等于原始空间的欧式距离，例如

$$\|\mathbf{z}_i - \mathbf{z}_j\| = dist_{i,j}, dist_{i,j} = d_{i,j}$$

- 如何计算 \mathbf{z}_i ？

高维空间点两两之间的距离矩阵尽可能地接近低维空间的距离矩阵。
问题看起来也有点像矩阵近似问题？
如果是矩阵近似问题，就可以用我们的老朋友SVD来得到。

➤ MDS的形式化

■不失一般性，我们假设低维空间中的样本是中心化的，即 $\sum_{i=1}^N \mathbf{z}_i = 0$ 。

■ $d_{i,j} = \|\mathbf{z}_i - \mathbf{z}_j\|$ ，所以： $d_{i,j}^2 = \|\mathbf{z}_i - \mathbf{z}_j\|^2 = \|\mathbf{z}_i\|_2^2 + \|\mathbf{z}_j\|_2^2 - 2\mathbf{z}_i^T \mathbf{z}_j$ 。

■对上述等式两边求和

$$\begin{aligned}\sum_{i=1}^N d_{i,j}^2 &= \sum_{i=1}^N \left(\|\mathbf{z}_i\|_2^2 + \|\mathbf{z}_j\|_2^2 - 2\mathbf{z}_i^T \mathbf{z}_j \right) \\ &= \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + N\|\mathbf{z}_j\|_2^2 - 2\mathbf{z}_j \underbrace{\sum_{i=1}^N \mathbf{z}_i}_{0} \\ &= \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + N\|\mathbf{z}_j\|_2^2\end{aligned}$$

➤ MDS的形式化

■类似的，得到

$$\sum_{i=1}^N d_{i,j}^2 = \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + N \|\mathbf{z}_j\|_2^2$$

$$\sum_{j=1}^N d_{i,j}^2 = \sum_{j=1}^N \|\mathbf{z}_j\|_2^2 + N \|\mathbf{z}_i\|_2^2$$

$$\sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 = \sum_{i=1}^N \left(\sum_{j=1}^N \|\mathbf{z}_j\|_2^2 + N \|\mathbf{z}_i\|_2^2 \right) = N \sum_{j=1}^N \|\mathbf{z}_j\|_2^2 + N \sum_{i=1}^N \|\mathbf{z}_i\|_2^2$$

$$= 2N \sum_{i=1}^N \|\mathbf{z}_i\|_2^2$$

➤ MDS的形式化

$$\sum_{i=1}^N d_{i,j}^2 = \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + N \|\mathbf{z}_j\|_2^2$$

$$\sum_{j=1}^N d_{i,j}^2 = \sum_{j=1}^N \|\mathbf{z}_j\|_2^2 + N \|\mathbf{z}_i\|_2^2$$

$$\sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 = 2N \sum_{i=1}^N \|\mathbf{z}_i\|_2^2$$

$$\frac{1}{N} \sum_{i=1}^N d_{i,j}^2 = \frac{1}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + \|\mathbf{z}_j\|_2^2$$

$$\frac{1}{N} \sum_{j=1}^N d_{i,j}^2 = \frac{1}{N} \sum_{j=1}^N \|\mathbf{z}_j\|_2^2 + \|\mathbf{z}_i\|_2^2$$

$$\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 = \frac{2}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2$$

➤ MDS的形式化

$$\frac{1}{N} \sum_{i=1}^N d_{i,j}^2 = \frac{1}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + \|\mathbf{z}_j\|_2^2$$

$$\frac{1}{N} \sum_{j=1}^N d_{i,j}^2 = \frac{1}{N} \sum_{j=1}^N \|\mathbf{z}_j\|_2^2 + \|\mathbf{z}_i\|_2^2$$

$$\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 = \frac{2}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2$$

$$\|\mathbf{z}_j\|_2^2 = \frac{1}{N} \sum_{i=1}^N (d_{i,j}^2 - \|\mathbf{z}_i\|_2^2)$$

$$\|\mathbf{z}_i\|_2^2 = \frac{1}{N} \sum_{j=1}^N (d_{i,j}^2 - \|\mathbf{z}_j\|_2^2)$$

$$\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 = \frac{2}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2$$

➤ 推导

■ 定义内积矩阵 $\mathbf{B} = \mathbf{Z}^T \mathbf{Z} \in \mathbb{R}^{N \times N}$, 即 $b_{ij} = \mathbf{z}_i^T \mathbf{z}_j$

$$d_{i,j}^2 = \|\mathbf{z}_i - \mathbf{z}_j\|^2 = \|\mathbf{z}_i\|_2^2 + \|\mathbf{z}_j\|_2^2 - 2\mathbf{z}_i^T \mathbf{z}_j = b_{i,i} + b_{j,j} - 2b_{i,j}$$

■ 用D表示B

$$b_{i,j} = -\frac{1}{2}(d_{i,j}^2 - b_{i,i} - b_{j,j})$$

$$= -\frac{1}{2}\left(d_{i,j}^2 - \frac{1}{N} \sum_{j=1}^N (d_{i,j}^2 - \|\mathbf{z}_j\|_2^2) - \frac{1}{N} \sum_{i=1}^N (d_{i,j}^2 - \|\mathbf{z}_i\|_2^2)\right)$$

$$= -\frac{1}{2}\left(d_{i,j}^2 - \frac{1}{N} \sum_{j=1}^N d_{i,j}^2 - \frac{1}{N} \sum_{i=1}^N d_{i,j}^2 + \frac{2}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2\right)$$

$$= -\frac{1}{2}\left(d_{i,j}^2 - \frac{1}{N} \sum_{j=1}^N d_{i,j}^2 - \frac{1}{N} \sum_{i=1}^N d_{i,j}^2 + \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2\right)$$

$$\begin{aligned}\|\mathbf{z}_j\|_2^2 &= \frac{1}{N} \sum_{i=1}^N (d_{i,j}^2 - \|\mathbf{z}_i\|_2^2) \\ \|\mathbf{z}_i\|_2^2 &= \frac{1}{N} \sum_{j=1}^N (d_{i,j}^2 - \|\mathbf{z}_j\|_2^2) \\ \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 &= \frac{2}{N} \sum_{i=1}^N \|\mathbf{z}_i\|_2^2\end{aligned}$$

➤ 推导

$$\blacksquare b_{i,j} = -\frac{1}{2} \left(d_{i,j}^2 - \frac{1}{N} \sum_{j=1}^N d_{i,j}^2 - \frac{1}{N} \sum_{i=1}^N d_{i,j}^2 + \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{i,j}^2 \right)$$

■ 用矩阵表示： $B = -\frac{1}{2}JDJ$, 其中 $J = I_N - \frac{1}{N}ee^T$ 为 **中心化矩阵**,

$$e = (1, 1, \dots, 1)^T$$

■ DJ 为从 D 中的每个元素里减去列均值， JDJ 的作用就是在此基础上再减去行均值，因此中心化矩阵的作用就是把元素的中心平移到坐标原点。

- 中心化过程相当于平移，并不会改变低维度点之间的距离。

➤ 中心化矩阵

- 对矩阵A ,

$$\frac{1}{N} \mathbf{A} \mathbf{e} \mathbf{e}^T = \frac{1}{N} \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \sum_{i=1}^N a_{1,i} & \sum_{i=1}^N a_{1,i} & \cdots & \sum_{i=1}^N a_{1,i} \\ \sum_{i=1}^N a_{2,i} & \sum_{i=1}^N a_{2,i} & \cdots & \sum_{i=1}^N a_{2,i} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^N a_{N,i} & \sum_{i=1}^N a_{N,i} & \cdots & \sum_{i=1}^N a_{N,i} \end{bmatrix}$$

第j行的每个元素都是A中第j行的均值

- 类似的 , $\frac{1}{N} \mathbf{e} \mathbf{e}^T \mathbf{A}$ 的第j列的每个元素都是A中第j列的均值。
- 因此定义中心化矩阵 : $\mathbf{J} = \mathbf{I}_N - \frac{1}{N} \mathbf{e} \mathbf{e}^T$
- $\mathbf{D}\mathbf{J}$ 为从D中的每个元素里减去列均值 , $\mathbf{J}\mathbf{D}\mathbf{J}$ 的作用就是在此基础上再减去行均值 , 因此中心化矩阵的作用就是把元素分布的中心平移到坐标原点。

➤ MDS

■ 对B做特征值分解：

$$\mathbf{B} = \mathbf{V}\Lambda\mathbf{V}^T$$

$$\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_D), \lambda_1 > \lambda_2 > \dots > \lambda_D > 0$$

\mathbf{V} 是对应的特征向量组成的矩阵，则

$$\mathbf{Z} = \Lambda^{1/2}\mathbf{V}^T \in \mathbb{R}^{D \times N}$$

■ 实际上, 我们使用 $D' \ll D$ 特征值矩阵 $\tilde{\Lambda} = diag(\lambda_1, \lambda_2, \dots, \lambda_{D'})$

$$\mathbf{Z} = \tilde{\Lambda}^{1/2}\tilde{\mathbf{V}}^T \in \mathbb{R}^{D' \times N}$$

➤ MDS算法过程

输入: 距离矩阵 $\mathbf{D} \in R^{N \times N}$

低维空间的维数: D'

算法过程:

- 计算 $d_{i,j}^2, \frac{1}{N} \sum_{j=1}^N d_{i,j}^2, \frac{1}{N} \sum_{i=1}^N d_{i,j}^2$
- 计算 $\mathbf{B} = (b_{i,j})$
- 对 \mathbf{B} 做特征值分解
- $\tilde{\Sigma}$ 是包含 D' 个最大特征值的对角矩阵, $\tilde{\mathbf{V}}$ 是特征向量组成的矩阵

输出: $\tilde{\Sigma}^{1/2} \tilde{\mathbf{V}}^T \in R^{D' \times N}$

➤ 线性降维的一般形式

- 数据点: $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N) \in \mathbb{R}^{D \times N}$
- 新空间的数据点 $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2 \dots \mathbf{z}_N) \in \mathbb{R}^{D' \times N}$
- 选择 D' 个方向向量 $\mathbf{W} = (\mathbf{w}, \mathbf{w}_2 \dots \mathbf{w}_{D'})$

$$\mathbf{W} = \begin{pmatrix} | & & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_{D'} \\ | & & | \end{pmatrix} \in \mathbb{R}^{D \times D'}$$

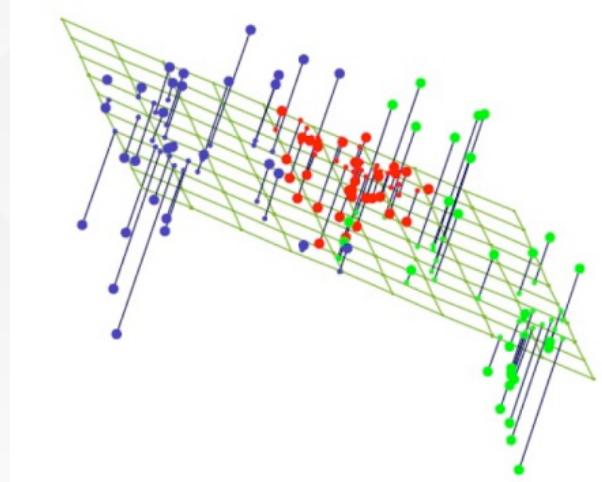
- 将 \mathbf{X} 投影到新空间 : $\mathbf{Z} = \mathbf{W}^T \mathbf{X}$ $\mathbf{x} \in \mathbb{R}^{361}$
 \downarrow $\mathbf{z} = \mathbf{W}^T \mathbf{x}$
 $\mathbf{z} \in \mathbb{R}^{10}$

如何选择 \mathbf{W} ?

➤ 主成分分析 (PCA) 目标函数1: 最小化重建误差

- 假定 $\|\mathbf{w}_i\| = 1, \mathbf{w}_i^T \mathbf{w}_j = 0(i \neq j)$, 为新的坐标系
- \mathbf{W} 用在两个函数中：
 - 编码： $\mathbf{z}_i = f(\mathbf{x}_i; \mathbf{W}) = \mathbf{W}^T \mathbf{x}_i, z_{ij} = \mathbf{w}_j^T \mathbf{x}_i$
 - 解码： $\hat{\mathbf{x}}_i = f^{-1}(\mathbf{z}_i; \mathbf{W}) = \mathbf{W}\mathbf{z}_i = \sum_{j=1}^{D'} z_{ij} \mathbf{w}_j$
- 我们希望让重建误差 $\|\mathbf{x} - \hat{\mathbf{x}}\|$ 尽可能小
- 目标：最小平方重建误差和

$$\min_{\mathbf{W} \in \mathbb{R}^{D \times D'}} \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{W}\mathbf{z}_i\|^2$$



▶ PCA 目标函数1：推导

$$\begin{aligned}
 & \sum_{i=1}^N \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \\
 &= \sum_{i=1}^N \left\| \sum_{j=1}^{D'} z_{i,j} \mathbf{w}_j - \mathbf{x}_i \right\|_2^2 \\
 &= \sum_{i=1}^N \mathbf{z}_i^T \mathbf{z}_i - 2 \sum_{i=1}^N \mathbf{z}_i^T \mathbf{W}^T \mathbf{x}_i + const
 \end{aligned}$$

$$\begin{aligned}
 & \|\mathbf{w}_i\|=1, \quad \mathbf{w}_i^T \mathbf{w}_j = 0(i \neq j) \\
 & \mathbf{z}_i = (z_{i,1}, z_{i,2} \dots z_{i,D'}) \\
 & \mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i
 \end{aligned}$$

$z_{i,j} = \mathbf{w}_j^T \mathbf{x}_i$ 是 \mathbf{x}_i 在低维坐标系下第 j 维坐标

$\hat{\mathbf{x}}_i = \sum_{j=1}^{D'} z_{i,j} \mathbf{w}_j$ 是基于 \mathbf{z}_i 重构得到的向量

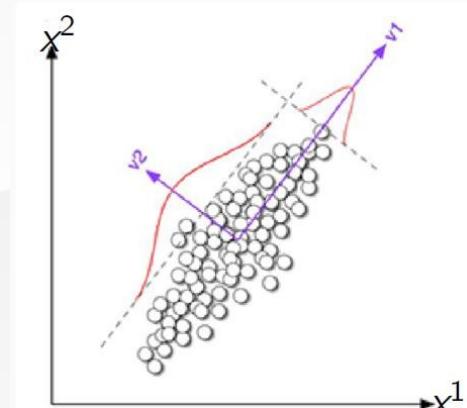
$$\begin{aligned}
 & \propto -tr \left(\mathbf{W}^T \left(\sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{W} \right) = -tr(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) = -tr(\mathbf{Z} \mathbf{Z}^T) = -\sum_{i=1}^N \mathbf{z}_i^T \mathbf{z}_i
 \end{aligned}$$

最小重构误差等价于最大投影后的方差

➤ 主成分分析 (PCA) 目标函数2: 最大投影后的方差

- 中心思想：寻找一个方向向量，使得数据投影到其上后方差最大
- 如果能找到，继续寻找下一个方向向量，该向量正交于之前的方向向量，并且尽可能保持数据方差
- 如果足够幸运，我们能够找到一些这样的方向向量，称为主成分，它们可以用来精确的描述数据。
- 目标是捕捉数据的内在变化
- 最大化样本投影的方差（假定样本已被中心化）

$$\max_{\mathbf{W}} \sum_i \mathbf{z}_i^T \mathbf{z}_i = \max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) = \max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{S} \mathbf{W})$$



▶ PCA目标函数计算

寻找 $\mathbf{W} \in \mathbb{R}^{D \times D'}$ 使得 : $\max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{S} \mathbf{W})$, s.t. $\mathbf{W}^T \mathbf{W} = \mathbf{I}_o$

■ 先计算第1个投影方向 :

- 假设将数据降至1维 , 此时投影矩阵只有一个列向量 \mathbf{w}_1
- 约束为 : $\mathbf{w}_1^T \mathbf{w}_1 = 1$,
- 目标函数为 $J = r(\mathbf{w}_1^T \mathbf{S} \mathbf{w}_1) = \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1$
- 拉格朗日函数 : $L = \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1 - \lambda_1 (\mathbf{w}_1^T \mathbf{w}_1 - 1)$
- $\frac{\partial L}{\partial \mathbf{w}_1} = 2\mathbf{S}\mathbf{w}_1 - 2\lambda_1 \mathbf{w}_1 = 0$,
- 则 $\mathbf{S}\mathbf{w}_1 = \lambda_1 \mathbf{w}_1$, 即 \mathbf{w}_1 是 \mathbf{S} 的特征向量。
- 目标函数 $J = \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1 = \mathbf{w}_1^T \lambda_1 \mathbf{w}_1 = \lambda_1$ 。
- 所以目标函数 J 要取最大值 , λ_1 是 \mathbf{S} 的最大的特征值。

➤ PCA目标函数计算

寻找 $\mathbf{W} \in \mathbb{R}^{D \times D'}$ 使得 : $\max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{S} \mathbf{W})$, s.t. $\mathbf{W}^T \mathbf{W} = \mathbf{I}_o$

■ 接着计算第2个投影方向 \mathbf{w}_2 :

- 此时投影矩阵有2个列向量 $\mathbf{w}_1, \mathbf{w}_2$
- 约束为 : $\mathbf{w}_2^T \mathbf{w}_2 = 1$, $\mathbf{w}_2^T \mathbf{w}_1 = 0$
- 目标函数为 $J = r(\mathbf{w}_1^T \mathbf{S} \mathbf{w}_1) = \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1$
- 拉格朗日函数 : $L = \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1 + \mathbf{w}_2^T \mathbf{S} \mathbf{w}_2 - \lambda_2 (\mathbf{w}_2^T \mathbf{w}_2 - 1) - \lambda_{2,1} \mathbf{w}_2^T \mathbf{w}_1$
- $\frac{\partial L}{\partial \mathbf{w}_2} = 2\mathbf{S}\mathbf{w}_2 - 2\lambda_2 \mathbf{w}_2 - \lambda_{2,1} \mathbf{w}_1 = 0$,
- 等式两边同乘以 , 得到 $2\mathbf{w}_1^T \mathbf{S} \mathbf{w}_2 - 2\lambda_2 \mathbf{w}_1^T \mathbf{w}_2 - \lambda_{2,1} \mathbf{w}_1^T \mathbf{w}_1 = 0 \Rightarrow \lambda_{2,1} = 0$
- 则 $\mathbf{S}\mathbf{w}_2 = \lambda_2 \mathbf{w}_2$, 即 \mathbf{w}_2 是 \mathbf{S} 的特征向量。
- 目标函数 $J = \lambda_1 + \lambda_2$ 。
- 所以目标函数 J 要取最大值 , λ_2 是 \mathbf{S} 的第2大的特征值。

➤ PCA目标函数计算

寻找 $\mathbf{W} \in \mathbb{R}^{D \times D'}$ 使得 : $\max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{S} \mathbf{W}), \ s.t. \quad \mathbf{W}^T \mathbf{W} = \mathbf{I}_o$

■ 依此类推 ,

- w_d 是 S 的特征向量 , 对应的特征值 λ_d 为 S 的第 d 大的特征值。

➤ 求解PCA

输入: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

低维空间的维度: D'

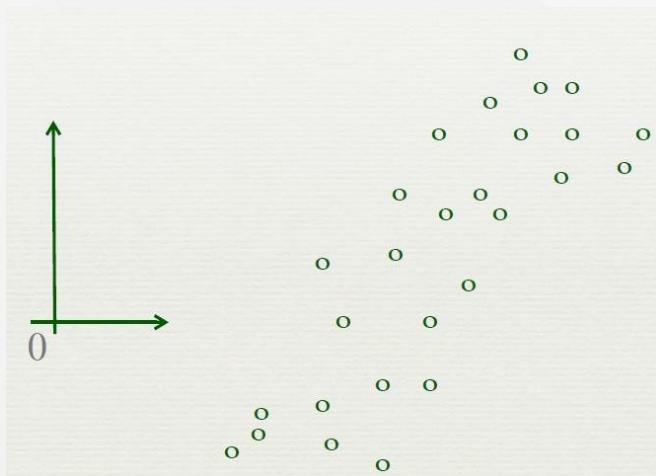
算法过程:

- $\mathbf{x}_i = \mathbf{x}_i - \frac{1}{N} \sum_{j=1}^N \mathbf{x}_j$
- 计算 $\mathbf{S} = \mathbf{X}\mathbf{X}^T$
- 对 \mathbf{S} 做特征分解
- D' 最大特征值对应的特征向量: $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{D'}$

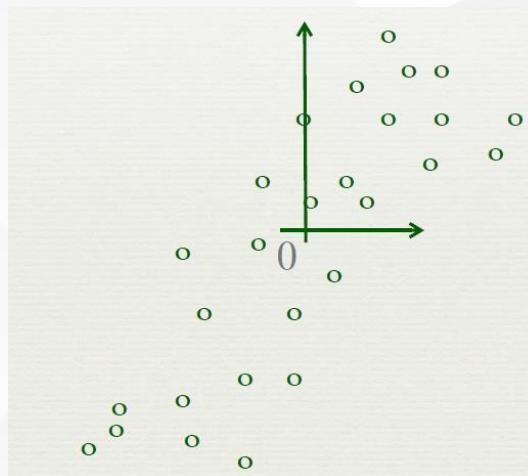
输出: $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{D'})$

求解PCA

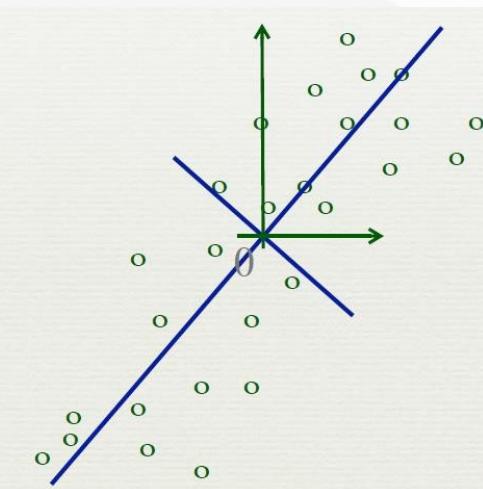
Original Data



Zero-centered data



Principle components



➤ 原始维度不是非常大时表现良好

- 如果原始维度非常大怎么办？

- e.g., Images ($D \geq 10^4$)

- 问题：协方差矩阵 \mathbf{S} 的形状为 $D \times D$

- $D = 10^4 \quad |\mathbf{S}| = 10^8$

- 奇异值分解(SVD)可以帮忙

- 非常高效的算法

- 有一些实现可以直接找到最大的 D 个特征值对应的特征向量

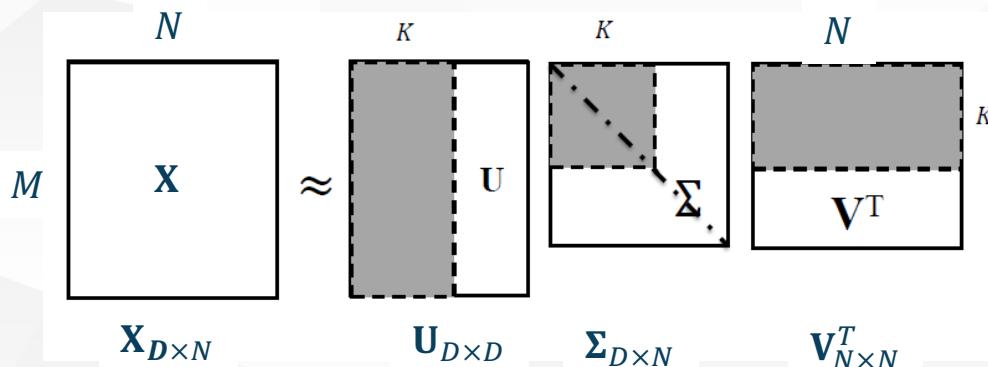
➤ 回顾: 奇异值分解 (SVD)

■前边我们看到主成成分是X的奇异值

■特别地:

- X的第d个主成成分就是X第d个左奇异值
- 第d个特征值就是第d个奇异值的平方(σ^2)

$$X = U \Sigma V^T$$



➤ 求解PCA(2)

输入: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_D\}$

低维空间的维数: D'

算法过程:

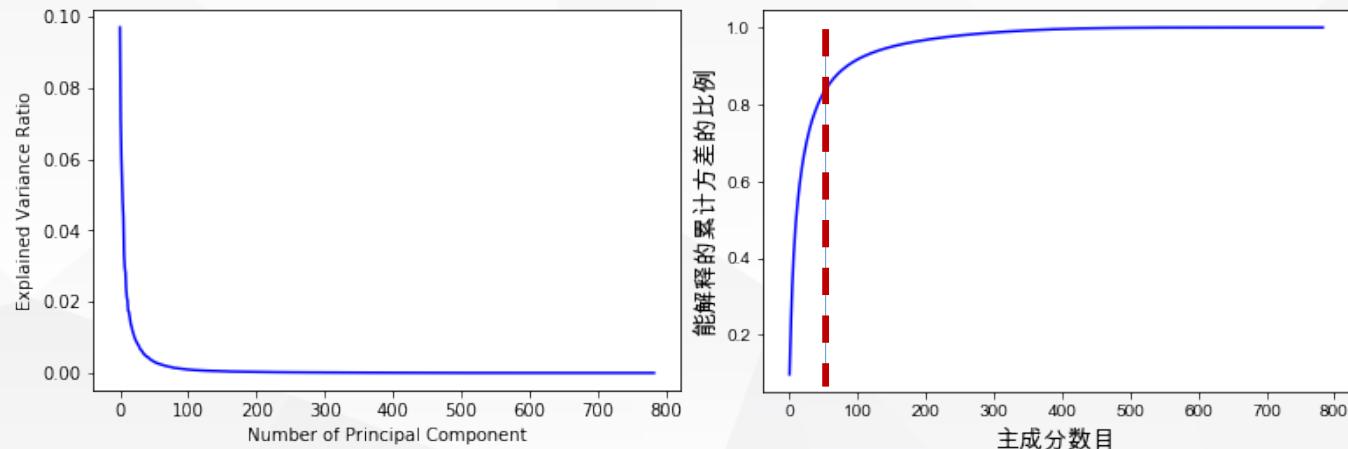
- $\mathbf{x}_i = \mathbf{x}_i - \frac{1}{N} \sum_{j=1}^N \mathbf{x}_j$
- 奇异值分解 $\mathbf{X} = \mathbf{U}_{D \times D} \boldsymbol{\Sigma}_{D \times N} \mathbf{V}_{N \times N}^T$
- D' 个奇异值对应的左奇异向量: $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{D'}$

输出: $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{D'})$

➤ 参数 D'

- 用户指定
- 利用简单分类器(如KNN) 通过交叉验证方式选择
- 或者使用重建阈值: $\sum_{d=1}^{D'} \lambda_d / \sum_{d=1}^D \lambda_d \geq t$ (e.g. 方差的85%)

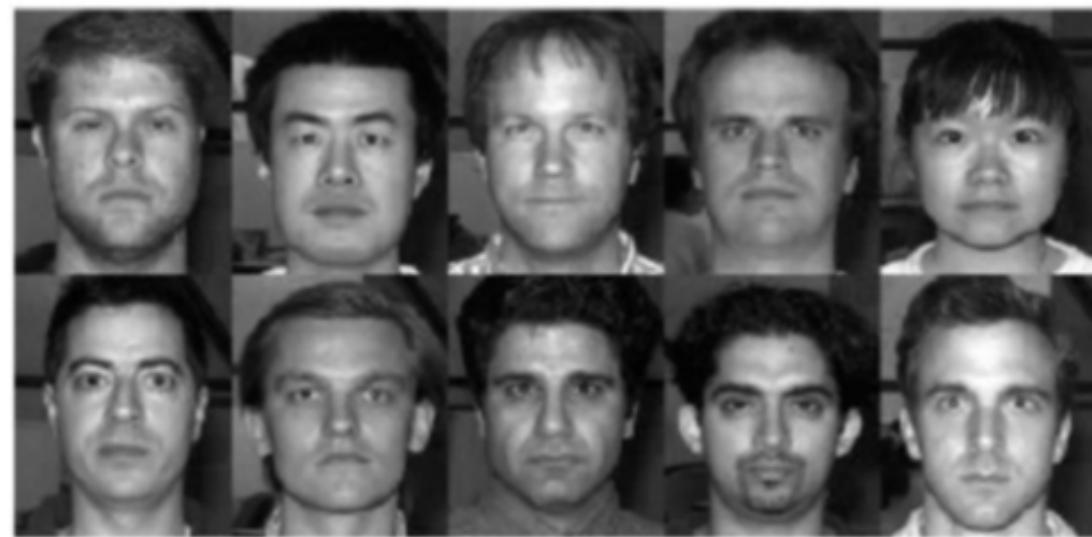
特征一般快速下降



MNIST数据集主成分数目
及对应方差之间的关系

➤ PCA应用 (1): 人脸识别

- 特征脸(Eigenfaces): 一组特征向量作为脸部图像的基本特征
- 计算特征脸:
 1. N 个人脸图像集合, 每个图像均表示为一个 D 维向量 $\mathbf{x}_i, \mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N\}$



➤ PCA用于人脸识别：特征脸

■ 计算特征脸：

2. 计算平均人脸图像 :: $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$

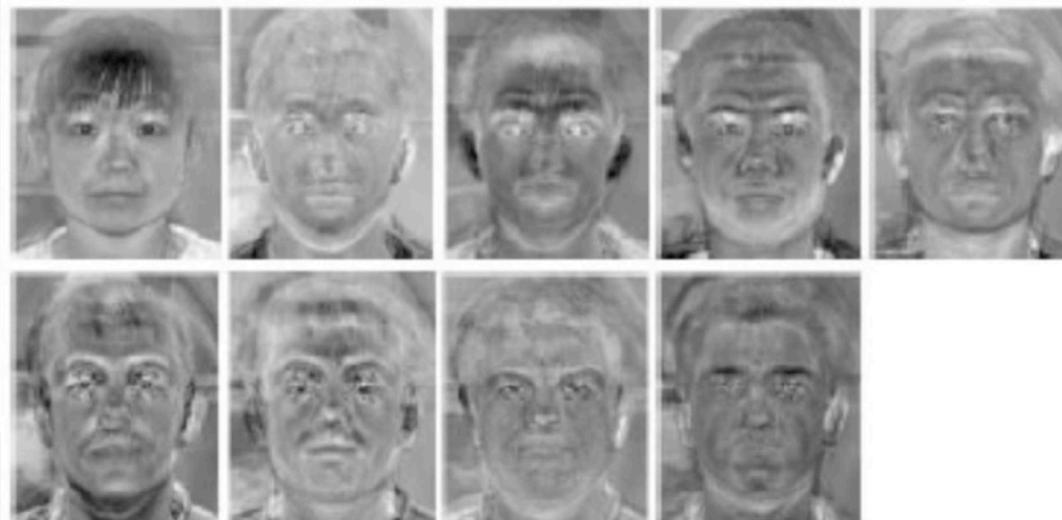


3. 减去平均人脸图像，得到： $\mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}}$, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N\}$

➤ PCA用于人脸识别: 特征脸

■ 计算特征脸:

4. 计算协方差矩阵 $S = XX^T$ 的特征值和特征向量。特征向量被称作特征脸，这些向量代表不同的人脸图片和均值脸最不相同的方向。



➤ PCA用于人脸识别: 特征脸

■ 计算特征脸:

5. 选择主成成分 例如. 选择 D' 个主成成分 , 根据

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_{D'}}{\lambda_1 + \lambda_2 + \cdots + \lambda_D} \geq t$$

■ 特征脸可以被用于表示已有的和新来的脸部图像 , 一个新的中心化的脸部头像可以被投影到特征量。

➤ PCA用于人脸识别: 特征脸

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + z_1 \mathbf{u}_1 + z_2 \mathbf{u}_2 + z_3 \mathbf{u}_3 \dots$$



Reconstruction Procedure



➤ PCA应用: 话题分析

■ 隐语义索引LSI (latent semantic indexing) 发现词语背后的含义；发现文档的主题

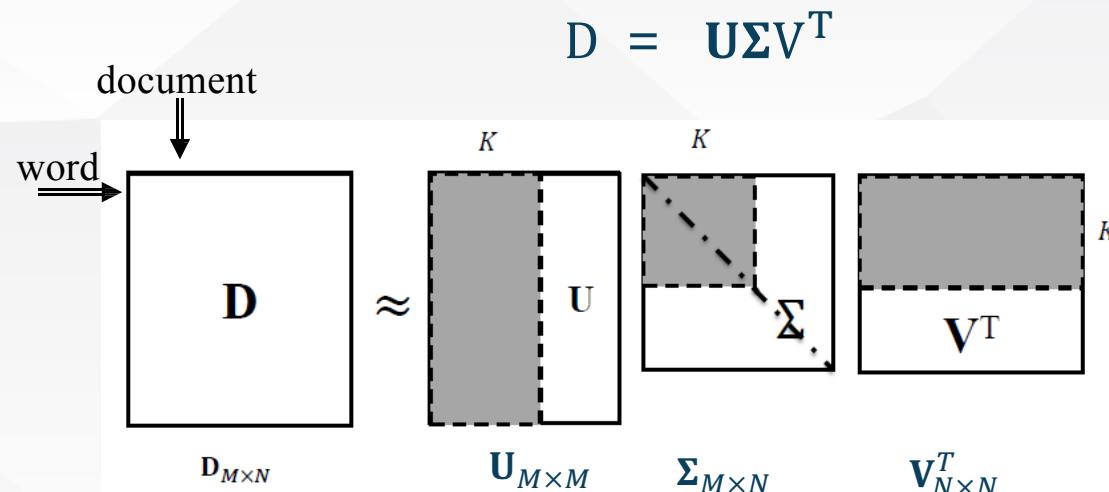
文档 – 单词 矩阵

Titles:
c1: Human machine interface for Lab ABC computer applications
c2: A survey of user opinion of computer system response time
c3: The EPS user interface management system
c4: System and human system engineering testing of EPS
c5: Relation of user-perceived response time to error measurement

m1: The generation of random, binary, unordered trees
m2: The intersection graph of paths in trees
m3: Graph minors IV: Widths of trees and well-quasi-ordering
m4: Graph minors: A survey

Terms	Documents								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

单词在文档中是否出现：0/1
单词在文档中出现的频率：自然数
单词在文档中TFIDF：实数



- $rank(U\Sigma_K V^T) = K \leq rank(D)$
- K 是隐含主题的个数
- u_k 代表第 k 个主题
- v_i^T 代表第 i 个文档用 K 个主题的表示

Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	Topic10
OPEC	Africa	contra	school	Noriega	firefight	plane	Saturday	Iran	senate
oil	South	Sandinista	student	Panama	ACR	crash	coastal	Iranian	Reagan
cent	African	rebel	teacher	Panamanian	forest	flight	estimate	Iraq	billion
barrel	Angola	Nicaragua	education	Delval	park	air	western	hostage	budget
price	apartheid	Nicaraguan	college	canal	blaze	airline	Minsch	Iraqi	Trade

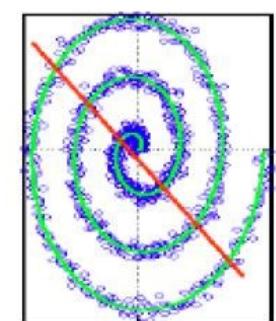
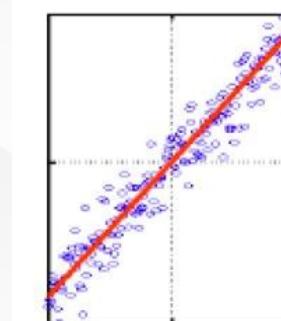
➤ PCA 总结

■ 优点:

- 特征向量方法
- 没有额外要调节的参数
- 没有迭代
- 没有局部最小值

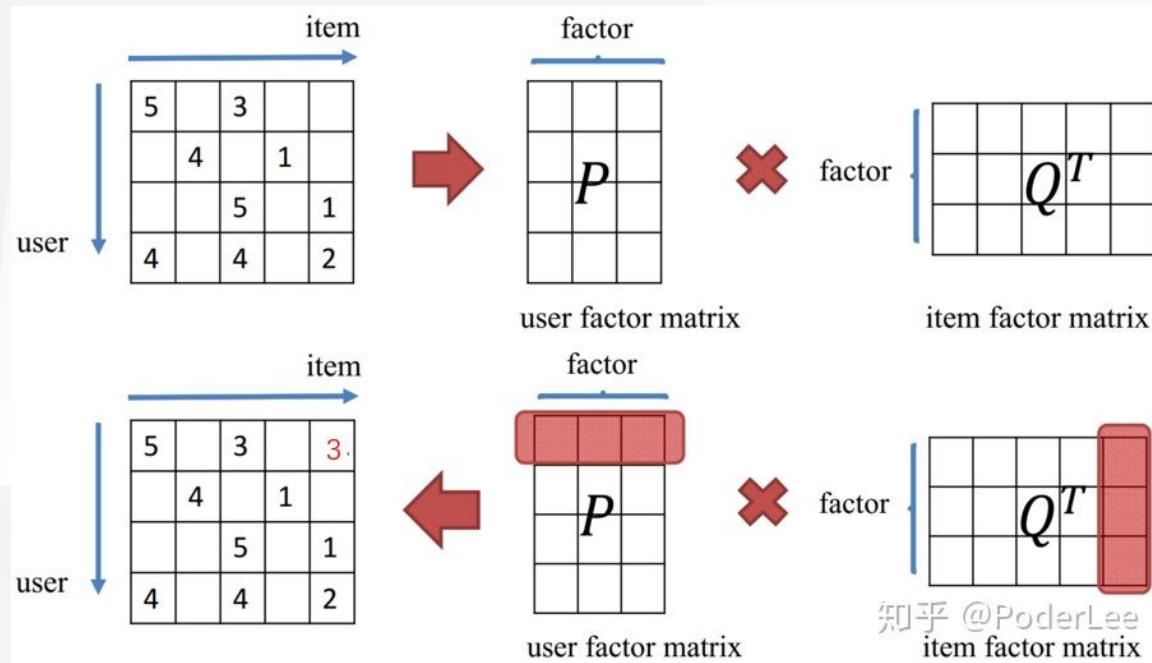
■ 弱点:

- 只用了二阶统计量，不能处理高阶依赖
- 受限于线性投影



➤ 矩阵分解

- 有时候存在两种对象，受到某种共同潜在因素（latent factor）的操控。
- 如果这些潜在因素，就可以对用户的行为进行预测，这也是**推荐系统**常用的方法之一。



- SVD分解只能处理稠密矩阵，推荐系统中的打分矩阵非常稀疏
- 因此最简单的办法是直接矩阵分解：

$$\mathbf{R}_{U \times I} = \mathbf{P}_{U \times K} \mathbf{Q}_{K \times I}^T$$

U ：用户数目， I ：物品数目，
 K ：隐含变量的数目

- 评分预测为：

$$\hat{r}_{u,i} = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^K p_{u,k} q_{i,k}$$

模型训练

- 利用R中的已知评分，训练P和Q，使得P和Q相乘的结果能最好地拟合已知的评分
- 目标函数为最小化总的误差平方和：

$$SSE = \frac{1}{2} \sum_{u,i} e_{ui}^2 = \frac{1}{2} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2 = \frac{1}{2} \sum_{u,i} \left(r_{ui} - \sum_{k=1}^K p_{uk} q_{ik} \right)^2$$

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

- 为避免过拟合，增加正则项，目标函数变为：

$$J(\theta) = \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2} \lambda \sum_u \|\mathbf{p}_u\|_2^2 + \frac{1}{2} \lambda \sum_i \|\mathbf{q}_i\|_2^2 = \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2} \lambda \sum_{u,k} p_{uk}^2 + \frac{1}{2} \lambda \sum_{i,k} q_{ik}^2$$

➤ 增加偏差项

- 用户对物品的打分不仅取决于用户和物品间的某种关系，
- 还取决于用户和物品独有的性质，预测模型变为：

$$\hat{r}_{u,i} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i$$

μ : 总的平均分

b_u : 用户 u 的属性值

b_i : 物品用户 i 的属性值

- 目标函数为：

$$J(\theta) = \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2}\lambda \sum_u \|\mathbf{p}_u\|_2^2 + \frac{1}{2}\lambda \sum_i \|\mathbf{q}_i\|_2^2 + \frac{1}{2}\lambda \sum_u b_u^2 + \frac{1}{2}\lambda \sum_i b_i^2$$

➤ 优化算法

- 随机梯度下降 (SGD)
- 交替最小二乘法 (Alternating Least Squares , ALS)
- 一般情况下随机梯度下降比ALS速度快。
- 但当系统能够并行化时，ALS的扩展性优于随机梯度下降法。

➤ 例：

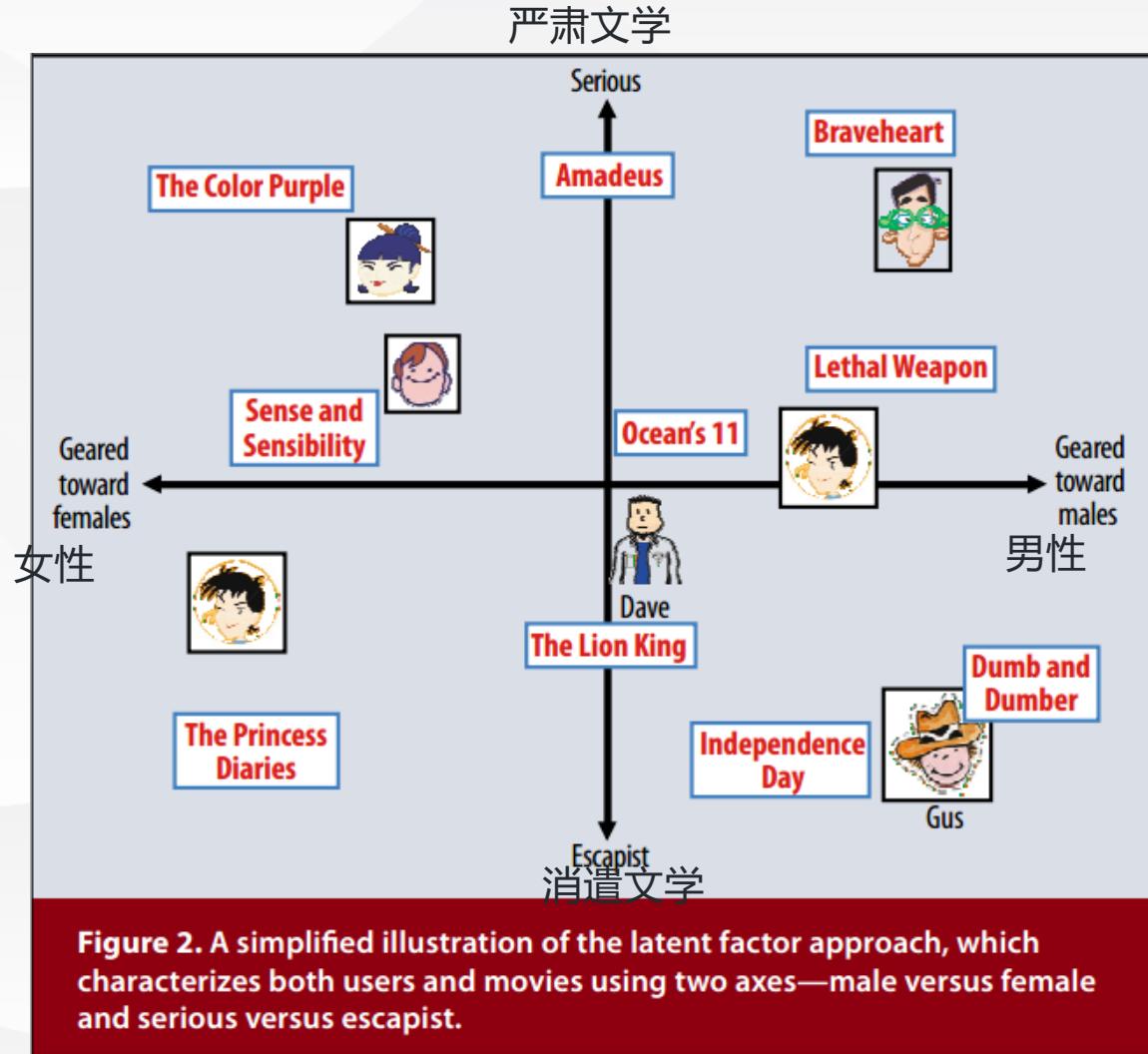


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

大纲

- 简介
- 降维算法
 - 维度选择
 - 维度抽取
 - 1. 预备知识
 - 2. 线性模型
 - 3. 非线性模型

➤ 非线性降维

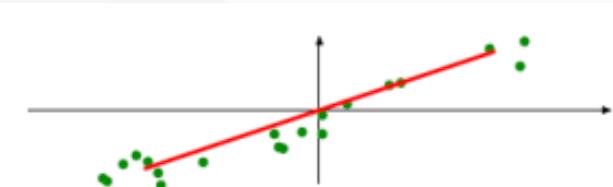
■ 基于重构：

- 核PCA (Kernelized PCA)
- 深度自编码器

■ 流形学习

- 全局距离保持：ISOMAP
- 局部距离保持：LLE、Laplacian Eigenmaps
- 局部优先，兼顾全局特征保存：T-NSE

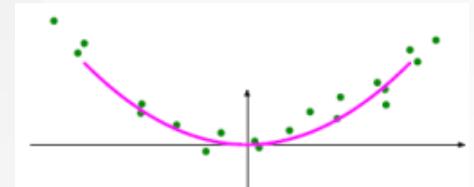
➤ 超越线性: 简单的解决方案



PCA 有效



PCA 无效



预期的解

- 对 x 进行非线性映射，然后再降维： $S = \{\phi(x) = Wz\}$

$\phi(x)$ 空间的线性降维



x 空间的非线性降维

- 问题：如何选择 $\phi(x)$

- (1) 启发式的解决方案（可能性太多）
- (2) $\phi(x)$ 很大，计算开销大

➤ 核PCA

- 表示理论：PCA 中，投影向量（主成分方向） \mathbf{w}_d 为 \mathbf{x}_i 的线性组合
- 回忆：PCA 转换成特征值问题： $\mathbf{X}\mathbf{X}^T\mathbf{w}_d = \lambda\mathbf{w}_d$

$$\mathbf{w}_d = \frac{1}{\lambda} \mathbf{X}\mathbf{X}^T\mathbf{w}_d = \frac{1}{\lambda} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T \mathbf{w}_d = \sum_{i=1}^N \mathbf{x}_i \frac{1}{\lambda} \mathbf{x}_i^T \mathbf{w}_d$$

注意 $\mathbf{w}_d = \mathbf{X}\boldsymbol{\alpha}_d = \sum_{i=1}^N \boldsymbol{\alpha}_{di} \mathbf{x}_i$ 其中 $\boldsymbol{\alpha}_d$ 为某种权重。

和 SVMs 相似： $\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$

- 关键事实：PCA 只需要内积： $\mathbf{K} = \mathbf{X}^T \mathbf{X}$

➤ 核PCA推导

- PCA 特征值问题： $\mathbf{X}\mathbf{X}^T \mathbf{w}_d = \lambda_d \mathbf{w}_d \Rightarrow \mathbf{w}_d = \mathbf{X}\boldsymbol{\alpha}_d$
- 将 $\mathbf{w}_d = \mathbf{X}\boldsymbol{\alpha}_d$ 代入，得到： $\mathbf{X}\mathbf{X}^T \mathbf{X}\boldsymbol{\alpha}_d = \lambda_d \mathbf{X}\boldsymbol{\alpha}_d$
- 两边同乘以 \mathbf{X}^T ，得到： $(\mathbf{X}^T \mathbf{X})(\mathbf{X}^T \mathbf{X})\boldsymbol{\alpha}_d = \lambda_d (\mathbf{X}^T \mathbf{X})\boldsymbol{\alpha}_d$
- 两边同乘以 $(\mathbf{X}^T \mathbf{X})^{-1}$ ，得到： $(\mathbf{X}^T \mathbf{X})\boldsymbol{\alpha} = \lambda_d \boldsymbol{\alpha}_d$
- 将内积矩阵 $(\mathbf{X}^T \mathbf{X})$ 用核矩阵 \mathbf{K} 表示，得到核PCA的目标： $\mathbf{K}\boldsymbol{\alpha}_d = \lambda_d \boldsymbol{\alpha}_d$ ，
 - 其中核矩阵元素 $k_{ij} = \mathbf{x}_i^T \mathbf{x}_j = k(\mathbf{x}_i, \mathbf{x}_j)$
 - 即对核矩阵 \mathbf{K} 进行特征值分解， $\boldsymbol{\alpha}_d$ 为特征值 λ_d 对应的特征向量。
- 将 $\boldsymbol{\alpha}_d$ 的代入 $\mathbf{w}_d = \mathbf{X}\boldsymbol{\alpha}_d$ ，可得到第 d 个投影方向， $d = 1, 2, \dots, D'$ 。

➤ 核PCA推导

■ 得到投影向量 $\mathbf{w}_d = \mathbf{X}\alpha_d$ 后，将 D' 个 \mathbf{w}_d 组合，得到投影矩阵：

$$\mathbf{W} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{D,1} & x_{D,2} & \cdots & x_{D,N} \end{pmatrix} \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,D'} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,D'} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N,1} & \alpha_{N,2} & \cdots & \alpha_{N,D'} \end{pmatrix} = \mathbf{XA}$$

■ 对任意样本 \mathbf{x} ，可得到其低维表示 \mathbf{z} 为：

$$\mathbf{z} = \mathbf{W}^T \mathbf{x} = (\mathbf{XA})^T \mathbf{x} = \mathbf{A}^T \mathbf{X}^T \mathbf{x} = \mathbf{A}^T \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ k(\mathbf{x}_2, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_N, \mathbf{x}) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^N \alpha_{1,i} k(\mathbf{x}_i, \mathbf{x}) \\ \sum_{i=1}^N \alpha_{2,i} k(\mathbf{x}_i, \mathbf{x}) \\ \vdots \\ \sum_{i=1}^N \alpha_{D',i} k(\mathbf{x}_i, \mathbf{x}) \end{pmatrix}, k(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T \mathbf{x}。$$

➤ KPCA 算法

■ 将上面推导中 $k(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T \mathbf{x}$ 换成任意核函数，就得到KPCA：

- 计算 \mathbf{K} , $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ ，获得中心化的 \mathbf{K} : $\bar{\mathbf{K}}$
- 对 $\bar{\mathbf{K}}$ 做特征分解
- 选择 D' 个最大的特征值对应的特征向量 $\alpha_1, \dots, \alpha_{D'}$ (α 为 N 维向量，通常对 α 进行归一化，模长为 1)

■ 对于新样本 \mathbf{x} ，可得到其低维表示 \mathbf{z} 为：

$$\mathbf{z} = \begin{pmatrix} \sum_{i=1}^N \alpha_{1,i} k(\mathbf{x}_i, \mathbf{x}) \\ \sum_{i=1}^N \alpha_{2,i} k(\mathbf{x}_i, \mathbf{x}) \\ \vdots \\ \sum_{i=1}^N \alpha_{D',i} k(\mathbf{x}_i, \mathbf{x}) \end{pmatrix}$$

对所有样本求和，开销大

➤ 非线性降维

- 核PCA (Kernelized PCA)
- 深度自编码器
- 流形学习
 - 全局距离保持 : ISOMAP
 - 局部距离保持 : LLE、Laplacian Eigenmaps
 - 局部优先，兼顾全局特征保存: T-NSE

➤ 流形学习

在高维空间中，欧氏距离不能准确地反映数据内在的相似性。

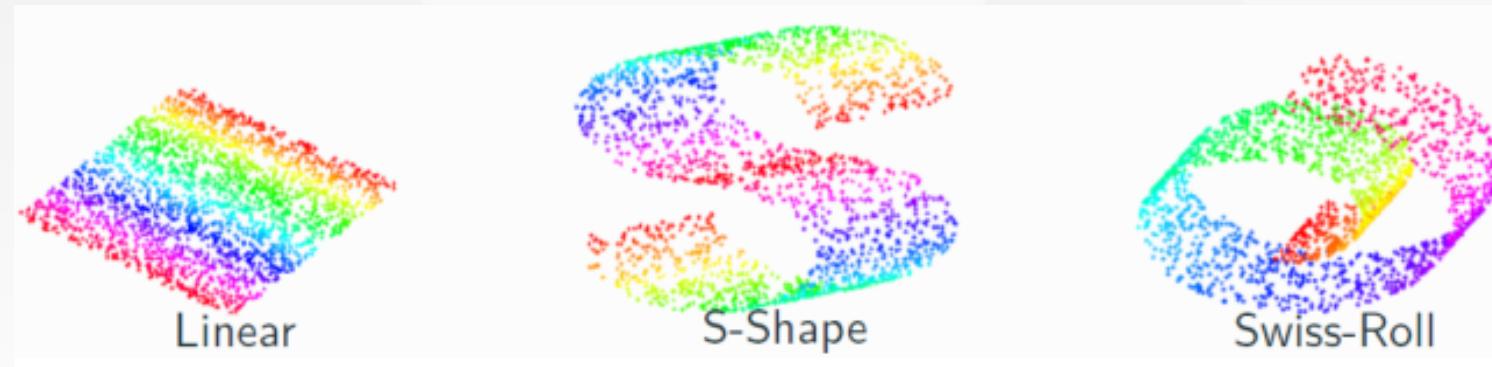
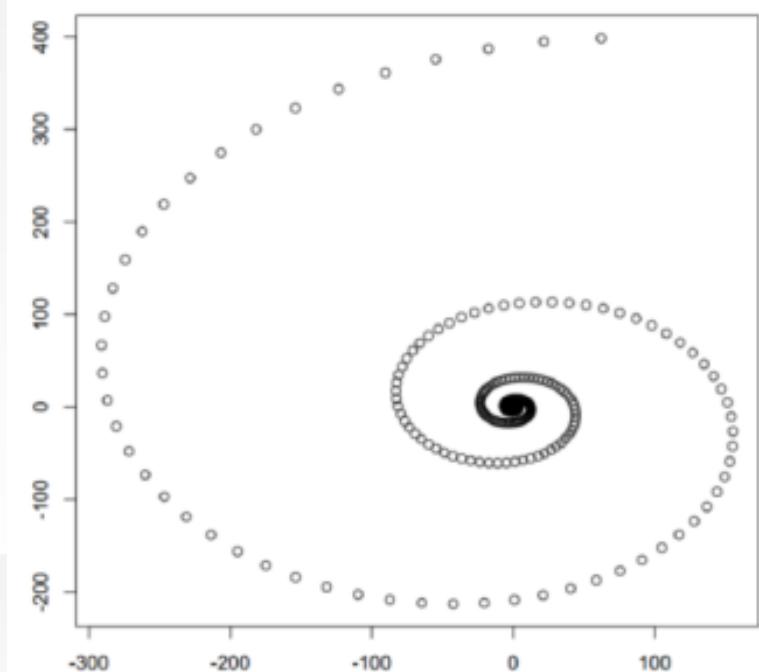


Figure 3: 2D data embedded into 3D space

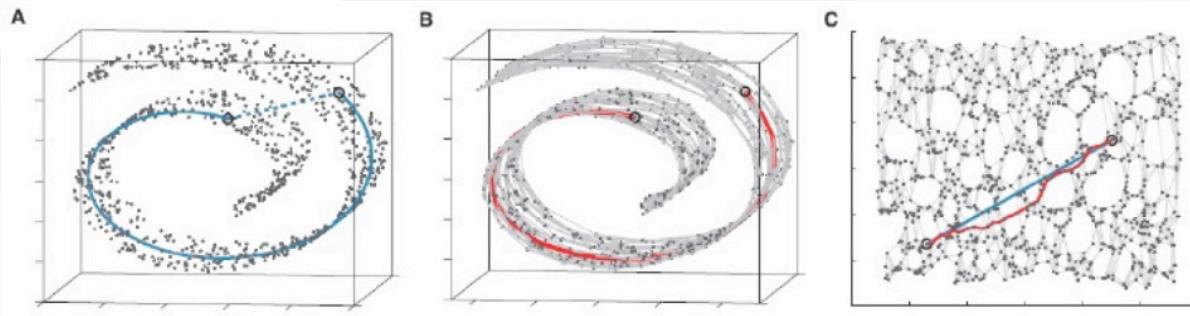
➤ 流形学习

- 流形是一个局部具有欧氏距离性质的拓扑空间。
- 流形能近似地任意高维空间的子空间。



➤ 测地线距离(Geodesic Distance)

■ 在高维空间中如何度量距离？



- 应该使用测地距离，而不是直接使用欧氏距离。
- 测地距离：
 - 邻近的点：输入空间的欧氏距离提供一个测地线距离的近似。
 - 较远的点：测点距离能够通过一系列邻域点之间的欧式距离的累加近似。

» ISOMAP是算法

- ISOMAP是一种**非线性**降维方法，基于度量MDS，试图保留数据内在的由测地距离蕴含的几何结构。
- ISOMAP是算法的三个步骤：
 - 构建邻接图
 - 计算最短路径（测地距离）
 - 构建低维嵌入

➤ ISOMAP算法

数据集 X 中任意两个点的距离为 $d_{i,j}$

■ 构建邻接图：

- 定义图 G ：两个点 i 和 j 距离小于 $\epsilon(\epsilon-\text{Isomap})$ ，或者 i 是 j 的 K 近邻之一(K -Isomap)，连接这两个点，且边的长度为 $d_{i,j}$ 。

■ 计算最短路径：

- 如果 i 和 j 相连接，初始化 $d_G(i,j) = d_{ij}$ ；否则 $d_G(i,j) = \infty$.
- 对于每一个 $k = 1, 2, \dots, N$ ，替换所有的 $d_G(i,j)$ 为 $\min(d_G(i,j), d_G(i,k) + d_G(k,j))$ ，则 $D_G = [d_G(i,j)]$ 包含 G 中所有点对的最短路径。

■ 通过MDS构建低维的数据嵌入（MDS只能得到样本点的坐标）

■ 对于新数据点，可以训练一个回归模型预测该点的低维表示。

- 高维空间中的坐标做为输入，低维空间中的坐标做为输出，回归输入与输出之间的关系。

» ISOMAP算法

■ ISOMAP算法中有两个瓶颈

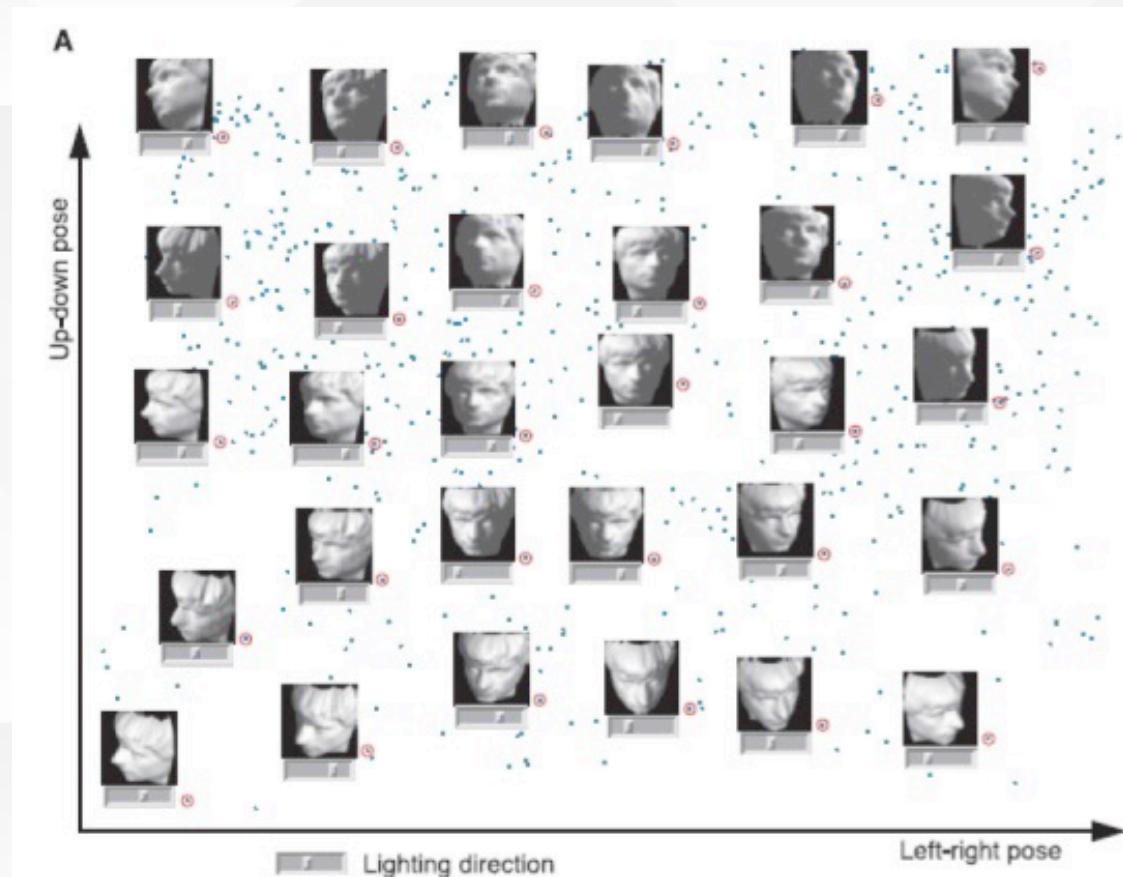
■ 最短路径的计算

- Floyd算法: $O(N^3)$

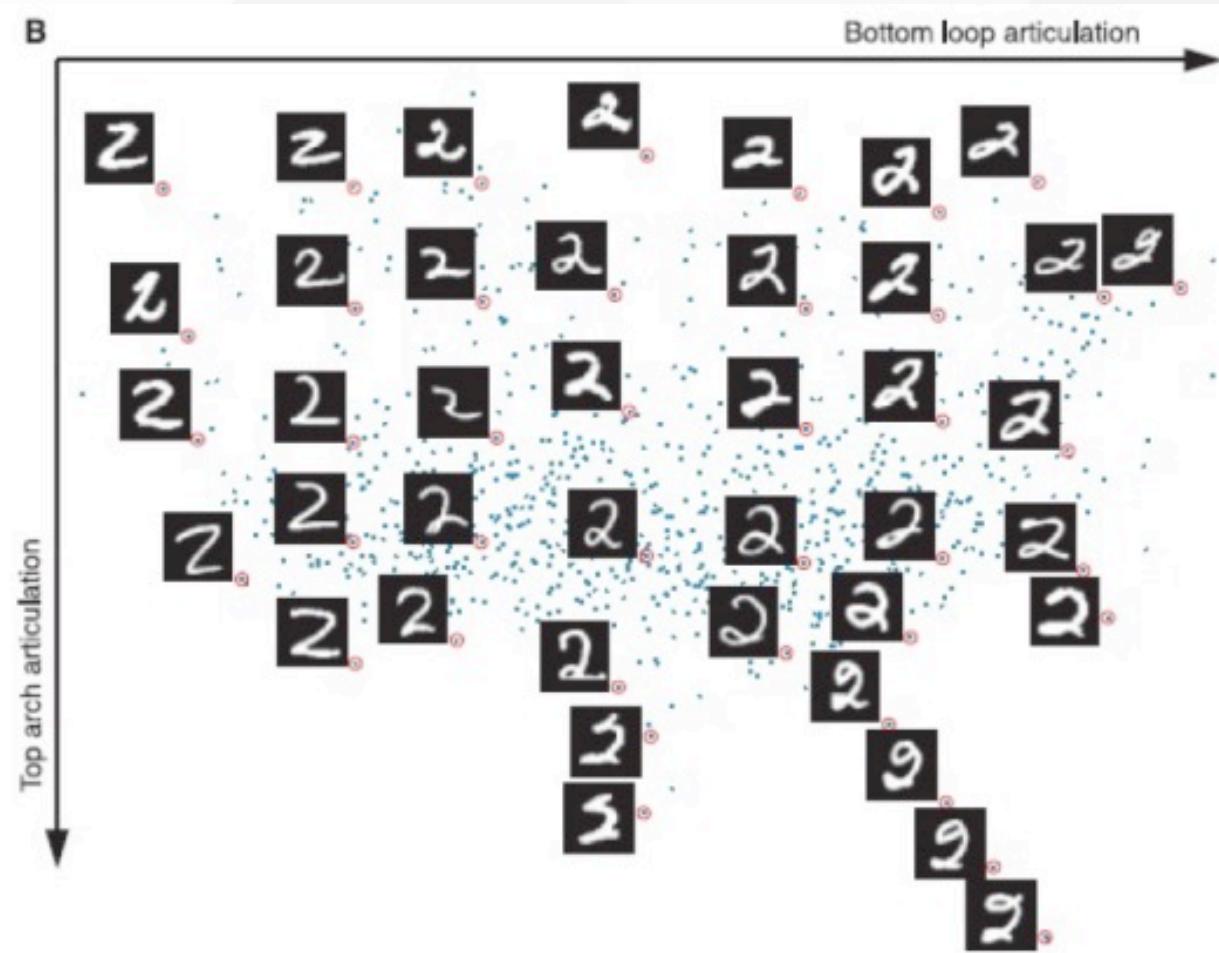
- Dijkstra算法(利用Fibonacci堆): $O(KN^2 \log N)$ 其中 K 为最近邻的个数

■ 特征分解: $O(N^3)$

➤ 例：人脸图像



➤ 例：数字“2” 图像

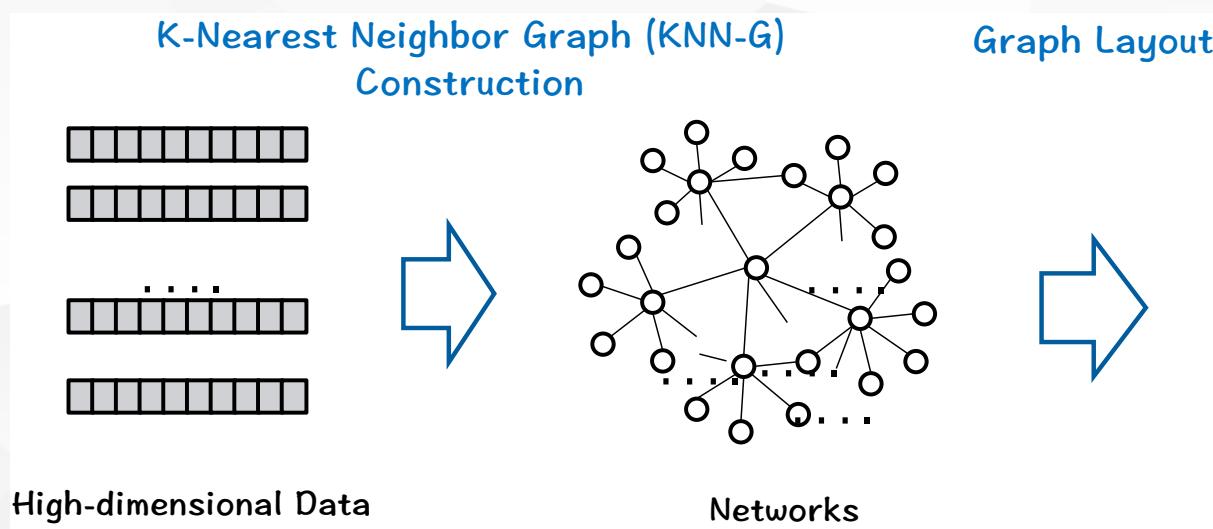


➤ 全局 vs. 局部嵌入方法

- MDS 或 ISOMAP 通过保持所有样本点对之间的（欧氏）距离或者测地距离来计算嵌入表达。因此他们属于全局嵌入方法。
- 局部线性嵌入 (LLE) 和拉普拉斯特征映射 (Laplacian eigenmap, LE) 试图通过保持局部的几何特性来恢复全局的非线性结构. 他们属于局部嵌入方法。
- T-SNE则是主要保存局部距离，但兼顾全局距离。

➤ 拉普拉斯特特征映射(Laplacian Eigenmap)

- 令 $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in \mathbb{R}^D$,
- 类似ISOMAP, 拉普拉斯特特征映射算法首先构建一个带权重的图表示邻接关系。
- 然后基于图计算**特征映射** (eigenmap) 。



➤ 边的创建

■ 如果 \mathbf{x}_i 和 \mathbf{x}_j 距离较近，则节点 i 和 j 连一条边。

■ 边创建的两个准则：

- ε 邻域 ($\varepsilon - neighborhood$) :

如果 $||\mathbf{x}_i - \mathbf{x}_j|| < \varepsilon$ ($\varepsilon \in \mathbb{R}^+$)， i 和 j 连一条边；

- K 近邻：

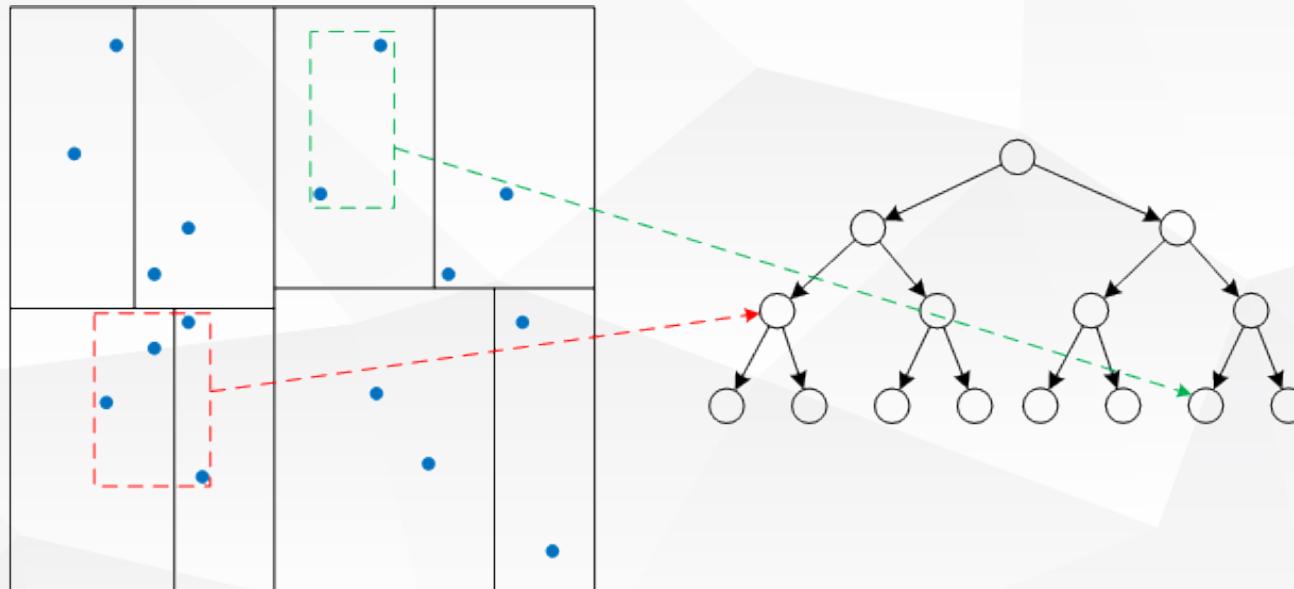
如果 \mathbf{x}_i 是 \mathbf{x}_j 的 K 个最近邻之一或者 \mathbf{x}_j 是 \mathbf{x}_i 的 K 个最近邻之一，则节点 i 和 j 连一条边。

➤ KNN图(K-Nearest Neighbour Graph)

- 空间分割树(space-partitioning trees)算法
 - K-D树和随机投影树
- 局部敏感哈希(locality sensitive hashing)算法
- 邻居搜索(neighbor exploring techniques)算法

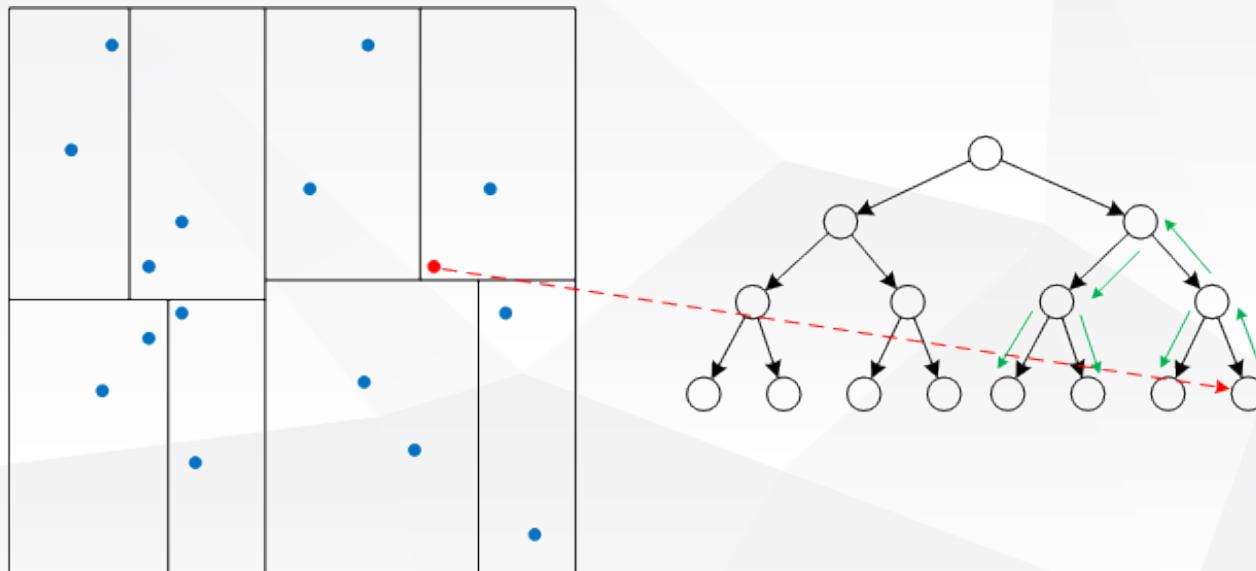
➤ K-d树

- K-d树是一种一棵二叉树，用于分割 K 维数据空间。
- 例：二维空间的K-d树，构建K-d树是一个递归的过程
 - 按照坐标轴方向划分空间（对高维数据，树的深度会很深）
 - 直到K-d树中所有叶子节点对应的点个数小于某个阈值



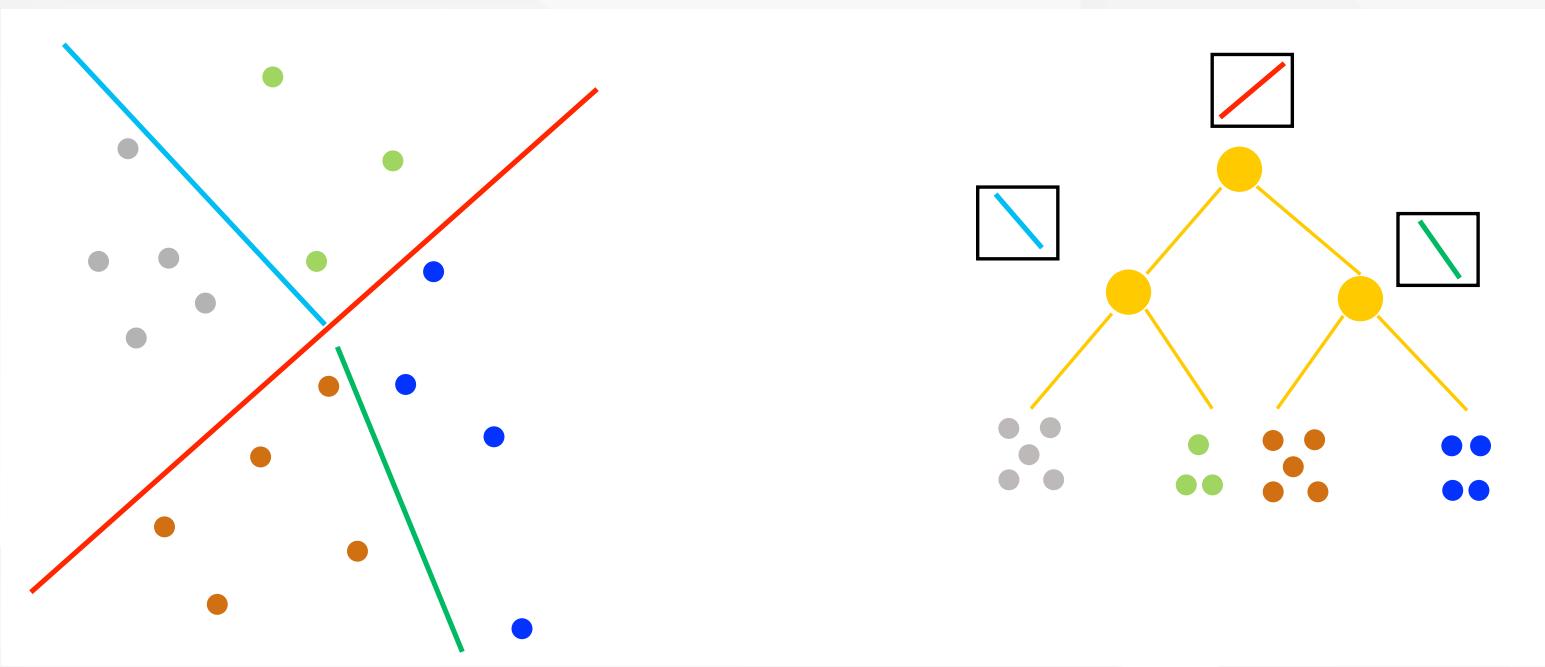
➤ K-d树 & M近邻

■ 寻找下图中红点的M近邻，只需要搜索当前子空间，同时不断回溯搜索父节点的其他子空间。



随机投影树

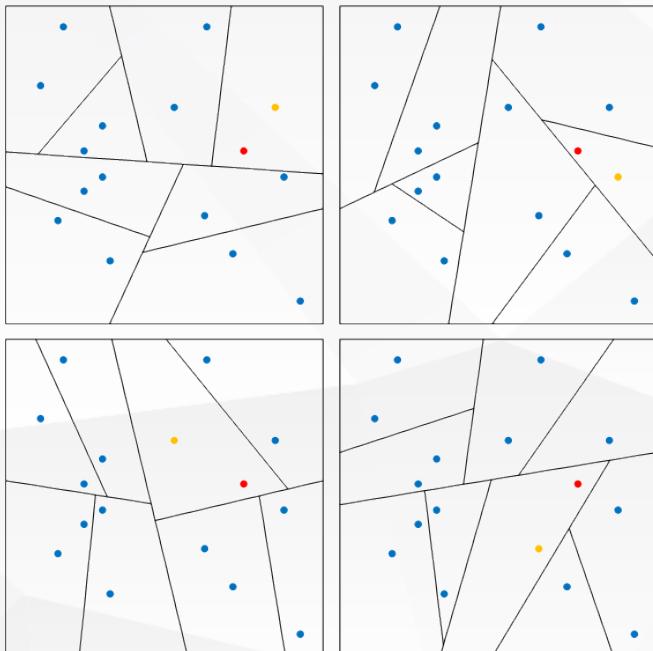
■ 基本思路与 K -d 树类似，但划分空间的方式不是按坐标轴，而是按随机产生的单位向量。



Learning the structure of manifolds using random projections

➤ 随机投影树 & M 近邻点

- 并行的构建多个随机投影树（如几百棵）
- 由于划分的单位向量都是随机产生的，因此每棵随机投影树对当前空间的划分都不相同



搜索红点的 M 近邻：在不同的随机投影树中搜索其所处的子空间，最后取并集即可。

还可以进一步考虑“邻居的邻居可能也是我的邻居”，利用邻居搜索算法寻找潜在的邻居，计算邻居与当前点、邻居的邻居与当前点的距离并放入一个小堆之中，取距离最小的 M 个节点作为 M 近邻，最终得到一个精确的 M NN图。

➤ 边权重

■ 两个常见的边权重计算方式:

- 热核 (Heat kernel) :

$$w_{i,j} = \begin{cases} \exp\left\{-\frac{\|x^{(i)} - x^{(j)}\|}{\sigma^2}\right\} & \text{如果 } i \text{ 和 } j \text{ 相连} \\ 0 & \text{otherwise} \end{cases}$$

其中 $\sigma^2 \in \mathbb{R}^+$

- 二值化权重 :

$$w_{i,j} = \begin{cases} 1 & \text{如果 } i \text{ 和 } j \text{ 相连} \\ 0 & \text{otherwise} \end{cases}$$

➤ 构建特征映射

- 如果上述方法构建的图不是连通的，那么下列步骤单独应用于每一个连通分量。
- 我们首先考虑寻找一维嵌入向量的特殊情况，然后泛化到一般的 D' 维的情况。
- 令 $\mathbf{z} = (z_1, z_2, \dots, z_N)^T$ 表示 1 维嵌入向量
- 需要最小化的目标函数如下

$$\sum_{i,j=1}^N (z_i - z_j)^2 w_{i,j}$$

主要思想：如果两个数据实例i和j很相似，那么i和j在降维后目标子空间中应该尽量接近。

➤ 构建特征映射

- 可以将目标函数写成如下形式

$$\begin{aligned}\frac{1}{2} \sum_{i,j=1}^N (z_i - z_j)^2 w_{i,j} &= \frac{1}{2} \sum_{i,j=1}^N z_i^2 w_{i,j} + \frac{1}{2} \sum_{i,j=1}^N z_j^2 w_{i,j} - \sum_{i,j=1}^N z_i z_j w_{i,j} \\&= \sum_{i=1}^N z_i^2 d_{i,i} - \sum_{i,j=1}^N z_i z_j w_{i,j} \\&= \sum_{i,j=1}^N z_i (d_{i,i} - w_{i,j}) z_j = \mathbf{z}^T (\mathbf{D} - \mathbf{W}) \mathbf{z} = \mathbf{z}^T \mathbf{L} \mathbf{z}\end{aligned}$$

其中 $d_{i,i} = \sum_j w_{i,j}$, $\mathbf{D} = diag(d_{1,1} \dots d_{N,N})$, $\mathbf{L} = \mathbf{D} - \mathbf{W}$ 是图的拉普拉斯矩阵。

➤ 尺度不变性

- 为了去掉嵌入向量中任意的尺度因子，我们添加约束 $\mathbf{z}^T \mathbf{Dz} = 1$ 。
- 优化问题可以被写成如下形式：

$$\min_{\mathbf{z}} \mathbf{z}^T \mathbf{Lz} \quad \text{s. t. } \mathbf{z}^T \mathbf{Dz} = 1$$

- 上述带约束的优化问题可用拉格朗日乘子法求解：

$$L = \mathbf{z}^T \mathbf{Lz} - \lambda(\mathbf{z}^T \mathbf{Dz} - 1)$$

- 拉格朗日函数对 \mathbf{z} 求偏导并等于0，得到 $\frac{\partial L}{\partial \mathbf{z}} = \mathbf{Lz} - \lambda \mathbf{Dz} = 0$ 。
- 因此 $\mathbf{Lz} = \lambda \mathbf{Dz}$ ，是一个广义特征值问题。通过求得最小非零特征值所对应的特征向量，即可达到降维的目的。

广义特征值问题

■对于形式如下的特征值问题：

■求数 λ ，使方程

$$Ax = \lambda Bx \quad (1)$$

■有非零解 x ，这里 A 为 n 阶实对称矩阵， B 为 n 阶实对称正定矩阵， x 为 n 维列向量，则称该问题为矩阵 A 相对于矩阵 B 的广义特征值问题，称满足上式要求的数 λ 为矩阵 A 相对于矩阵 B 的特征值，而与 λ 相对应的非零解 x 称为属于 λ 的特征向量。

■由于 B 正定，故用 B^{-1} 左乘(1)式两端得： $B^{-1}Ax = \lambda x$

■这样可将广义特征值问题转化为矩阵 $B^{-1}A$ 的普通特征值问题。

➤ 尺度不变性

- 因此 $Lz = \lambda Dz$ ，是一个广义特征值问题。通过求得最小非零特征值所对应的特征向量，即可达到降维的目的。
- 最小：将 $Lz = \lambda Dz$ 代回目标函数 $\min_z z^T Lz \text{ s.t. } z^T Dz = 1$ ，得到
$$z^T Lz = z^T \lambda Dz = \lambda z^T Dz = \lambda$$
- $z^T Lz$ 最小，所以 λ 最小。

➤ 构建 $D' > 1$ 特征映射

- 我们用 $N \times D'$ 矩阵 $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N)^T$ 表示 D' 维嵌入后的表达
- 注意 \mathbf{z}_i 是 \mathbf{x}_i 在嵌入空间的 D' 维表示
- 需要最小化的目标函数如下

$$\sum_{i,j=1}^N \|\mathbf{z}_i - \mathbf{z}_j\|^2 w_{i,j}$$

➤ 构建 $D' > 1$ 特征映射

- 最小化的问题可以被写成如下形式

$$\min_{\mathbf{z}} \text{tr}(\mathbf{Z}^T \mathbf{L} \mathbf{Z})$$

$$\text{s.t. } \mathbf{Z}^T \mathbf{D} \mathbf{Z} = \mathbf{I}, \mathbf{Z}^T \mathbf{D} \mathbf{I} = \mathbf{0}.$$

- 形式类似PCA的优化问题。
- 解决方法：解如下泛化的特征分解问题，得到 D' 个最小的特征值对应的特征向量 $\mathbf{z}_k (k = 1 \dots D')$ 给出上述优化问题的解

$$\mathbf{L} \mathbf{z} = \lambda \mathbf{D} \mathbf{z}$$

- 特征向量进行归一化，使得对于所有的 $k = 1 \dots D'$ ， $\mathbf{z}_k^T \mathbf{D} \mathbf{z}_k = 1$ 成立。

➤ T-SNE

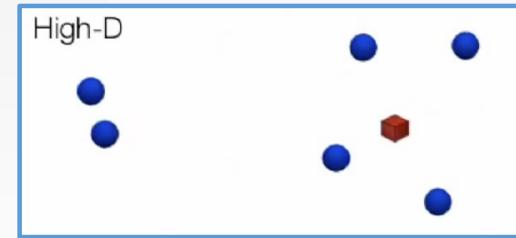
- T-SNE (T-distributed Stochastic Neighbor Embedding) 是一种非线性降维算法（流形学习算法），非常适用于高维数据降维到2维或者3维，进行可视化。
- 基本思想：在高维空间相似的数据点，映射到低维空间也相似。
 - SNE：将欧氏距离转换为用概率来表示的相似性。
- T-SNE：原始空间中的相似度由高斯联合概率表示，嵌入空间的相似度由“学生T分布”表示。

L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. JMLR, 2008.
L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. JMLR, 2014.

➤ SNE : Stochastic Neighbor Embedding

- 给定高维空间中的 N 个数据点： $\mathbf{x}_1, \dots, \mathbf{x}_N$
- 高维空间中的两个数据点 \mathbf{x}_j 和 \mathbf{x}_i ，之间的相似度用以数据点 \mathbf{x}_i 为
中心点的高斯分布表示：

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma_i^2)}{\sum_{j' \neq i} \exp(-\|\mathbf{x}_{j'} - \mathbf{x}_i\|^2 / 2\sigma_i^2)}$$

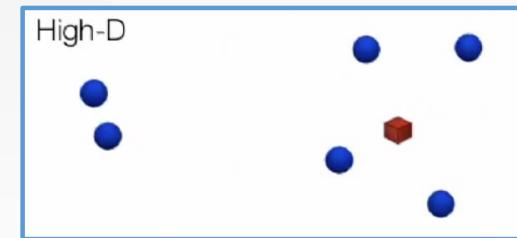


- 其中参数 σ_i 为以数据点 \mathbf{x}_i 为中心点的高斯分布的标准差。
- 复杂度： $O(N \log N)$ （只计算最邻居之间的相似度）

➤ 对称SNE

- 给定高维空间中的 N 个数据点： $\mathbf{x}_1, \dots, \mathbf{x}_N$
- 高维空间中的两个数据点 \mathbf{x}_j 和 \mathbf{x}_i 之间的相似度用以数据点 \mathbf{x}_i 为中心点的高斯分布表示：

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma_i^2)}{\sum_{j' \neq i} \exp(-\|\mathbf{x}_{j'} - \mathbf{x}_i\|^2 / 2\sigma_i^2)}$$



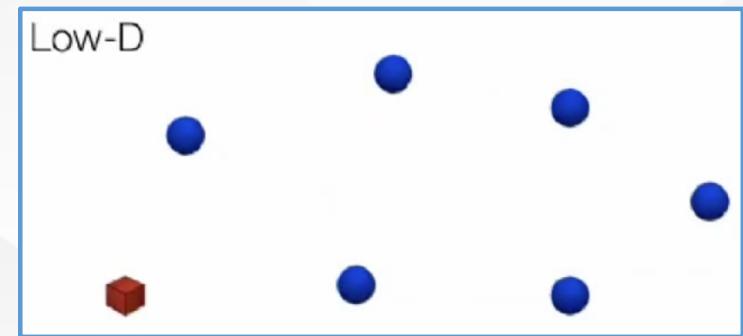
- 对称SNE：数据点 \mathbf{x}_j 和 \mathbf{x}_i 之间的相似度用联合分布表示：

$$p_{i,j} = \frac{p_{j|i} + p_{i|j}}{2N}$$

➤ t-SNE

- 将高维映射到低维空间，得到对应点为： $\mathbf{z}_1, \dots, \mathbf{z}_N$
- 低维空间中的两个对应数据点 \mathbf{z}_j 和 \mathbf{z}_i 之间的相似度自由度为1的**t分布**重新定义表示：

$$q_{i,j} = \frac{\left(1 + \|\mathbf{z}_j - \mathbf{z}_i\|^2\right)^{-1}}{\sum_{j' \neq i} \left(1 + \|\mathbf{z}_{j'} - \mathbf{z}_i\|^2\right)^{-1}}$$



➤ 目标函数

- 若 z_j 和 z_i 真实反映了高维数据点 x_j 和 x_i 之间的关系，那么概率 p_{ij} 与 q_{ij} 应该完全相等。
- 用 **KL散度**(Kullback-Leibler Divergence)衡量两个分布之间的距离。
t-SNE 的目标函数就是对所有数据点，最小化 KL 散度：

$$C = KL(P||Q) = \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$$

- 注意：KL 散度不是凸函数，具有不同初始值的多次运行将收敛于 KL 散度函数的局部最小值中，会获得不同的结果。因此，可尝试不同的随机数种子，并选择具有最低 KL 散度值的结果。

➤ 目标函数

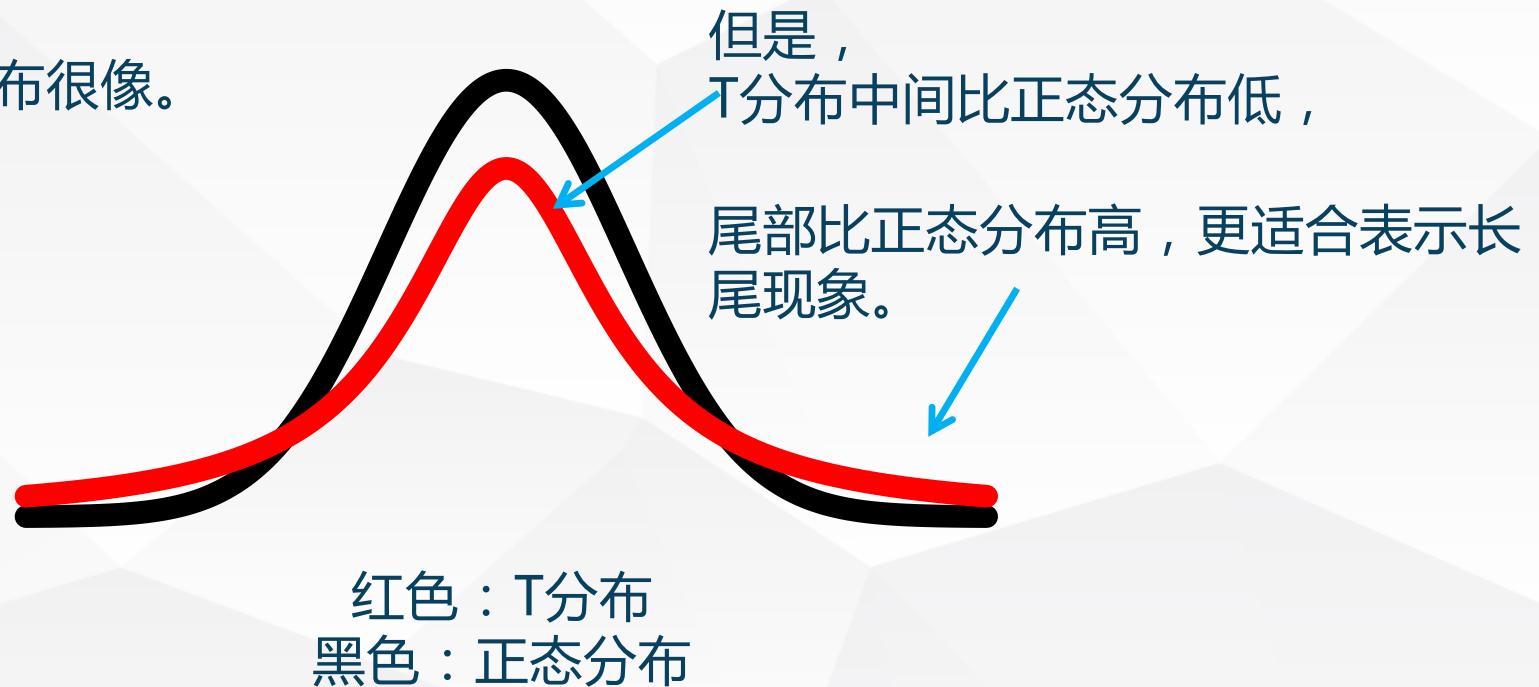
- KL散度(Kullback-Leibler Divergence)：

$$J = KL(P||Q) = \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$$

- KL散度具有不对称性：在低维映射中不同的距离对应的惩罚权重不同。
 - 当高维空间中两个点的距离较近， $p_{i,j}$ 较大；如果在低维空间中对应两个点的距离较远， $q_{i,j}$ 较小，则此时费用很大。（例如： $p_{i,j} = 0.8$ ， $q_{ij} = 0.2$ ， $C = 1.11$ ）
 - 当高维空间中两个点的距离较远， $p_{i,j}$ 较小；如果在低维空间中对应两个点的距离较近， q_{ij} 较大，则此时费用较小。（例如： $p_{i,j} = 0.2$ ， $q_{i,j} = 0.8$ ， $C = -0.277$ ）
 - SNE的代价函数更关注局部结构，T分布的使用会兼顾全局结构。

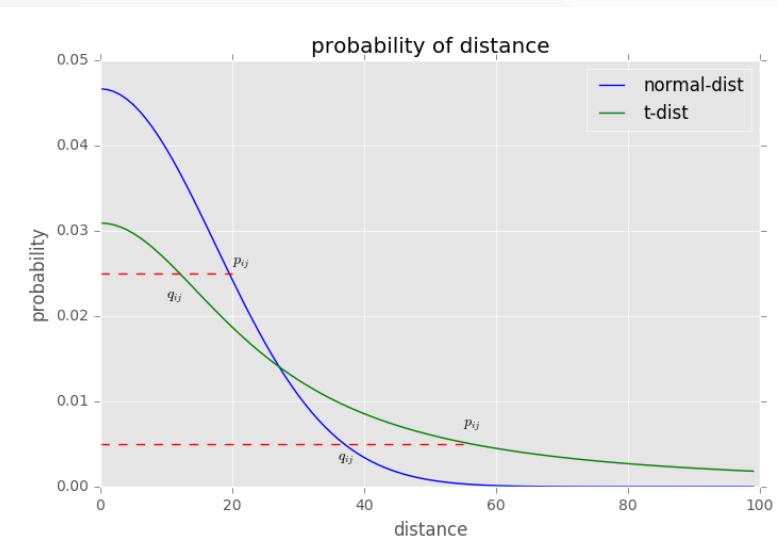
» 为什么用T分布？

T分布与正态分布很像。



➤ 为什么用t分布？

- 降维后局部结构保持：高维空间中距离小的点对，如果概率 $q_{i,j} = p_{i,j}$ ，在低维空间中距离更小。（下图中上面的红虚线）
- 兼顾全局结构保持：在高维空间距离较大的点对，如果概率 $q_{i,j} = p_{i,j}$ ，在低维空间中距离可以更大。T分布是长尾分布，允许有大距离的点对出现。（下面的红色虚线）



扩展比较密的区域
压缩稀疏区域

高斯分布对应高维空间，T分布对应低维空间

目标函数的优化求解

- 目标函数为 $J = KL(P||Q) = \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}$
- 其中 $q_{i,j} = \frac{(1 + \|z_j - z_i\|^2)^{-1}}{\sum_{j' \neq i} (1 + \|z_{j'} - z_i\|^2)^{-1}}$
- 目标函数优化求解可采用梯度下降法。
梯度 : $\frac{\partial J}{\partial z_i} = 4 \sum_j (p_{i,j} - q_{i,j}) (1 + \|z_i - z_j\|^2)^{-1} (z_i - z_j)$


➤ 优化动态结果

9

➤ 进阶话题

- 参数 σ_i (T-NSE中用困惑度表示) 的选择和影响
- 梯度的快速近似算法

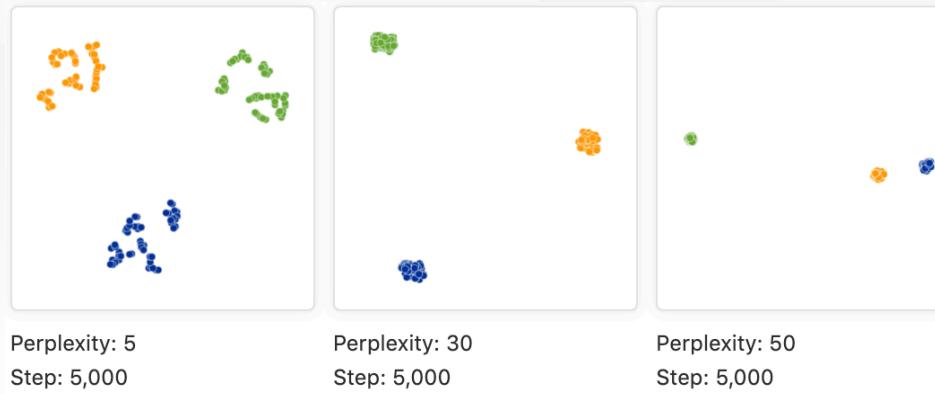
➤ T-SNE超参数的影响

■混淆度 (Perplexity) : 每个点的局部邻居的数目

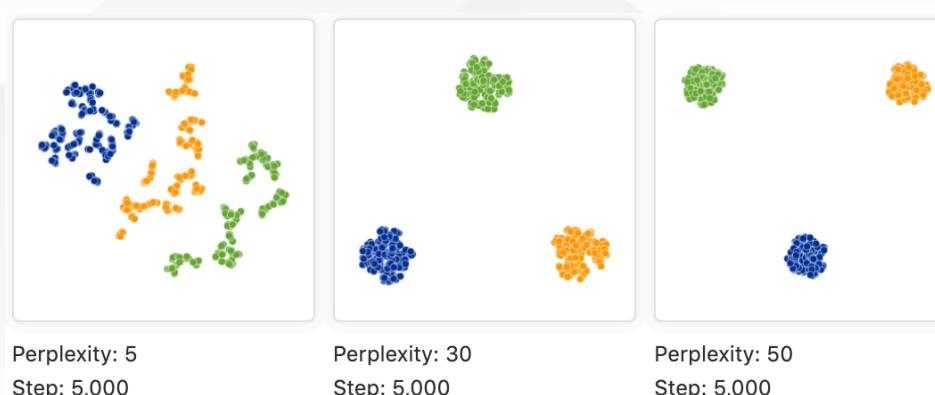
- 局部结构和全局结构之间折中

<https://distill.pub/2016/misread-tsne/>

原始输入，
每个簇50个点



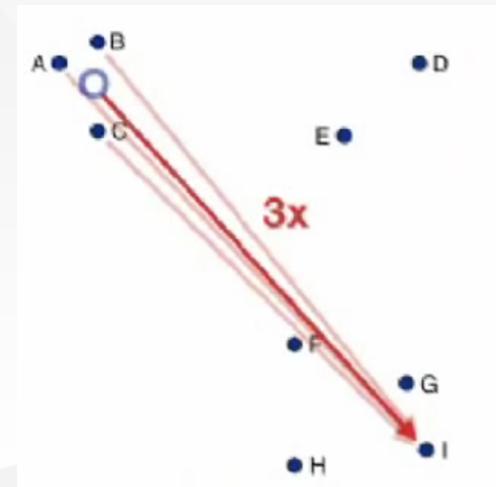
原始输入，
每个簇200个点



大数据集混淆度
应该设得更大

➤ Barnes-Hut 近似算法

- 目标函数梯度计算 : $\frac{\partial J}{\partial \mathbf{z}_i} = 4 \sum_j (p_{i,j} - q_{i,j}) (1 + \| \mathbf{z}_i - \mathbf{z}_j \|^2)^{-1} (\mathbf{z}_i - \mathbf{z}_j)$
 - 对所有数据点对求和 : $O(N^2)$
- Barnes-Hut 算法 :
 - 相似点的影响用同一个值代替
 - 只考虑邻近点的作用 (更远点的概率值小)
 - 用四叉树实现 ($O(uN \log N)$)
- 速度更快 , 可以处理更多的数据 (如数十万个样本)。



u : 混淆度

➤ 用KNN图表示高维空间中点的相似性

■ 用高斯分布描述高维空间中整体的距离分布关系：

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2 / 2\sigma_i^2)}{\sum_{j' \neq i} \exp(-\|\mathbf{x}_{j'} - \mathbf{x}_i\|^2 / 2\sigma_i^2)}$$

■ 当数据量 N 较大时，分母中 $\sum_{j' \neq i}$ 的计算量非常大。

■ 实际上，两个相距较远的点互为邻居的概率 $p_{j|i}$ 非常小，可以忽略不计。

■ 因此，在对一个点构建距离相似性关系时，不必考虑图中的每一个节点，只需考虑与其相近的若干个节点（ K 近邻）即可。

➤ 用KNN图表示高维空间中点的相似性

■ 考虑与点 \mathbf{x}_i 最近的 $3K$ 个点，其中 u 为点 \mathbf{x}_i 的周围条件概率分布的perplexity，将这些点的集合表示为 \mathcal{N}_i ，可得到

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2/2\sigma_i^2)}{\sum_{j' \in \mathcal{N}_i} \exp(-\|\mathbf{x}_{j'} - \mathbf{x}_i\|^2/2\sigma_i^2)}$$

- 这样大大降低了计算量。
- 但也引入了一个新的问题，我们必须先构建一个高维空间的KNN图。

➤ 梯度

■ 目标函数梯度计算：

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{z}_i} &= 4 \sum_j (p_{ij} - q_{ij})(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2)^{-1} (\mathbf{z}_i - \mathbf{z}_j) \\&= 4 \sum_j (p_{ij} - q_{ij})q_{ij}Z(\mathbf{z}_i - \mathbf{z}_j) = 4 \left(\underbrace{\sum_j p_{ij}q_{ij}Z(\mathbf{z}_i - \mathbf{z}_j)}_{\text{两个点之间的引力}} - \underbrace{\sum_j q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)}_{\text{两个点之间的斥力}} \right)\end{aligned}$$

- 引力部分计算起来较为容易，因为 $q_{ij}Z(z_i - z_j) = (1 + \|z_i - z_j\|^2)^{-1}$ ，可以采用K近邻近似时间复杂度为 $O(uN)$ 。
- 但是计算斥力依然有较大的计算量，时间复杂度约为 $O(N^2)$
- 因为 P 和 Q 的分布不同，可以使用一些优化技巧来简化计算。

➤ 斥力计算

- 假设有三个点 \mathbf{z}_i , \mathbf{z}_j , \mathbf{z}_k , 当 $\mathbf{z}_i - \mathbf{z}_j \approx \mathbf{z}_i - \mathbf{z}_k \gg \mathbf{z}_j - \mathbf{z}_k$ 时, 可以认为点 \mathbf{z}_j 和点 \mathbf{z}_k 对 \mathbf{z}_i 的斥力是近似相等的。



- 一片区域中每个点对 \mathbf{z}_i 的斥力均可用同一个值来近似。
- 问题：任意给定一个点，如何快速搜索出符合条件的区域？

➤ 斥力计算

■问题：任意给定一个点，如何快速搜索出符合条件的区域？

- 用四叉树来完成区域搜索任务
- Barnes-Hut算法搜索并验证符合近似条件的点-区域对

■进一步，考虑一个区域与另一个区域之间斥力的近似。

- 需要判断两个区域之间的斥力是否满足近似的条件
- 搜索并验证符合近似条件的区域-区域对（ Dual-tree 算法 ）



➤ Barnes-Hut 近似算法

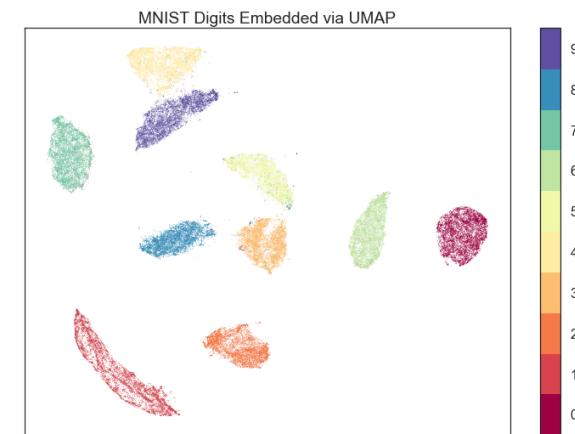
- Barnes-Hut近似算法在速度上做了优化，可以处理更多的数据，但是需要注意：
 - Barnes-Hut仅在目标维度为3或更小时才起作用，以2D可视化为主。
 - Barnes-Hut仅适用于密集的输入数据。稀疏数据矩阵只能用特定的方法嵌入，或者可以通过投影近似，例如使用 [sklearn.decomposition.TruncatedSVD](#)。
 - Barnes-Hut使用angle参数对近似进行控制。

➤ T-SNE的缺点

- T-SNE的计算复杂度很高，在数百万个样本数据集中可能需要几个小时，而PCA可以在几秒钟或几分钟内完成。 Barnes-Hut近似算法只限于二维或三维嵌入。
- 算法是随机的，具有不同种子的多次实验可以产生不同的结果。虽然选择费用最小的结果就行，但可能需要多次实验以选择超参数。
- 全局结构未明确保留。这个问题可以通过PCA初始化点来缓解。
- 主要用于可视化，很难用于其他目的，如对测试集的降维。

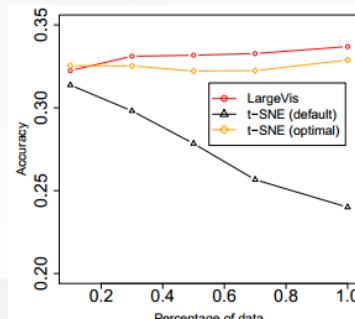
■统一流形逼近和投影 (Uniform Manifold Approximation and Projection , UMAP)

- <https://github.com/lmcinnes/umap>
- 类似T-SNE
- 但不仅仅用于可视化，还可以降维
- 更快
- 可对测试样本生成低维表示
- 基于 K 最近邻的概念，计算高维空间中点之间的距离
- 投影到低维空间中，计算低维空间中点之间的距离
- 目标：这些距离之间的差异最小
- 优化：随机梯度下降，可处理更多样本

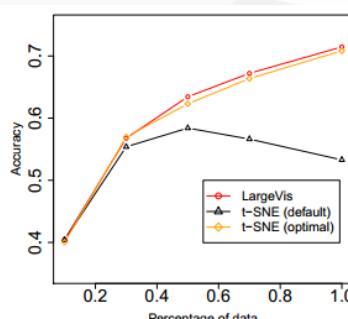


➤ LargeVis : 更快的T-NSE

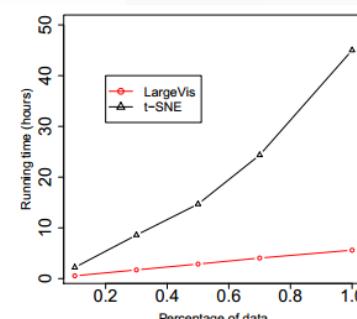
- Visualizing Large-scale and High-dimensional Data
 - 主页 (ppt和代码) : <https://sites.google.com/site/pkujiantang/big-data-visualization>
 - 更高效KNN图构建算法
 - 采用负采样和边采样优化
 - 采用异步随机梯度下降训练模型，在稀疏图上非常有效



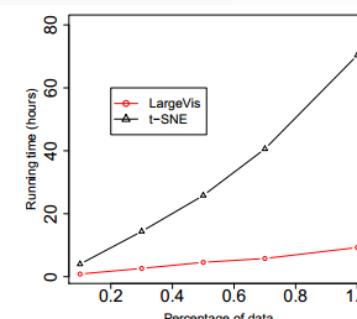
(a) Accuracy (WikiDoc)



(b) Accuracy (LiveJournal)



(c) Time (WikiDoc)

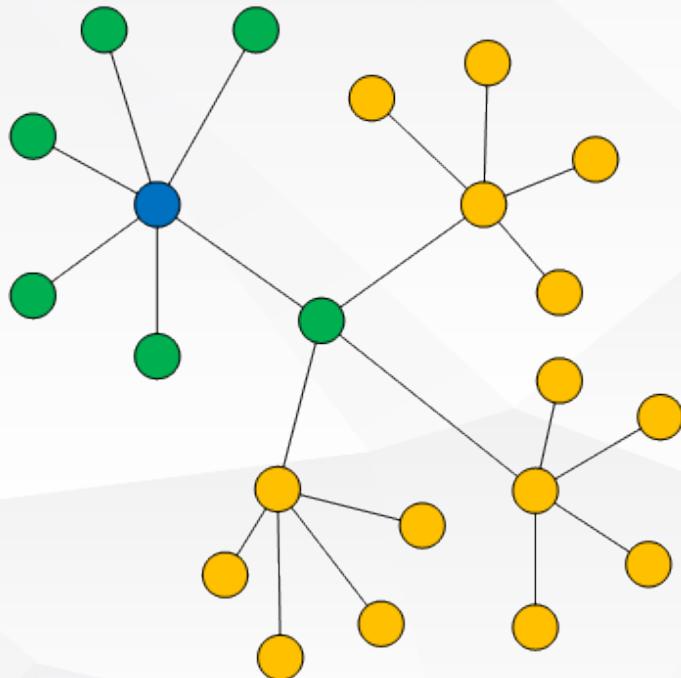


(d) Time (LiveJournal)

- Jian Tang, Jingzhou Liu, Ming Zhang and Qiaozhu Mei. [Visualizing Large-scale and High-dimensional Data](#). In WWW'16.

➤ 负采样

- 正样本：节点及其邻近节点构成一个正样本
- 负样本：中心点与非邻居点构成一个负样本



若蓝点为中心点，
正样本：每个绿点可与蓝点构成正样本
负样本：每个黄点与蓝点构成负样本

低维映射：在低维空间中，正样本中的节点对聚合在一起，负样本中的节点对分散的较远。

➤ 负采样

■先考虑无权值网络的情况， \mathbf{z}_i ， \mathbf{z}_j 表示低维空间中的两个点，两个点在KNN图中有一条二元边 $e_{i,j} = 1$ (权值为1的边)的概率为：

$$P(e_{i,j} = 1) = f(\|\mathbf{z}_i - \mathbf{z}_j\|^2)$$

■其中 $f()$ 用T-SNE里的T分布，即 $f(x) = \frac{1}{1+x^2}$ 。

•若 \mathbf{z}_i ， \mathbf{z}_j 之间的距离越小，两点在KNN图中有二元边的概率较大。

■LargeVis还考虑了有权值网络的情况，定义边权值为 $w_{i,j}$ 的概率为：

$$P(e_{i,j} = w_{i,j}) = P(e_{i,j} = 1)^{w_{i,j}}$$

➤ 负采样

■ 假定正样本集合为 E ，负样本集合为 \bar{E}

- 正样本和负样本的集合都可以直接通过kNN图获得

■ 优化目标可以写为：

$$O = \prod_{(i,j) \in E} P(e_{i,j} = 1)^{w_{i,j}} \prod_{(i,j) \in \bar{E}} (1 - P(e_{i,j} = 1))^\gamma$$

■ 最大化正样本的节点对在KNN图中有连接边的概率

■ 最小化负样本的节点对在KNN图中有连接边的概率，其中 γ 是负样本边的权值

➤ 负采样

■ 对优化目标 $O = \prod_{(i,j) \in E} P(e_{i,j} = 1)^{w_{i,j}} \prod_{(i,j) \in \bar{E}} (1 - P(e_{i,j} = 1))^\gamma$

■ 取一个对数，优化目标变为：

$$O = \sum_{(i,j) \in E} w_{i,j} \log(P(e_{i,j} = 1)) + \sum_{(i,j) \in \bar{E}} \gamma \log(1 - P(e_{i,j} = 1))$$

■ 由于目标函数需要对所有负样本进行计算，计算量太大：**负采样**

■ 对每一个点 i ，根据一个噪声分布 $P_n(j)$ ，随机选取 M 个点与 i 构成负样本。噪声分布可采用 $P_n(j) \propto d_j^{0.75}$ ，其中 d_j 为点 j 的度。

➤ 正采样

■ 根据负采样重新定义目标函数：

$$O = \sum_{(i,j) \in E} w_{i,j} \log(P(e_{i,j} = 1)) + \sum_{k=1}^M E_{j_k \sim Pn(j)} \gamma \log(1 - P(e_{i,j_k} = 1)) = 1$$

■ 对上述目标函数可采用随机梯度下降法求解。

■ 但求出梯度后，边权值 $w_{i,j}$ 仍是作为乘积项出现

• 边的权值 $w_{i,j}$ 变化范围很大。受 $w_{i,j}$ 的影响，梯度的变化也较大，对训练常不利

■ 正样本采样：正样本中两个点之间边的权值为 $w_{i,j}$ ，可将其转换为 $w_{i,j}$ 个重叠的二元边

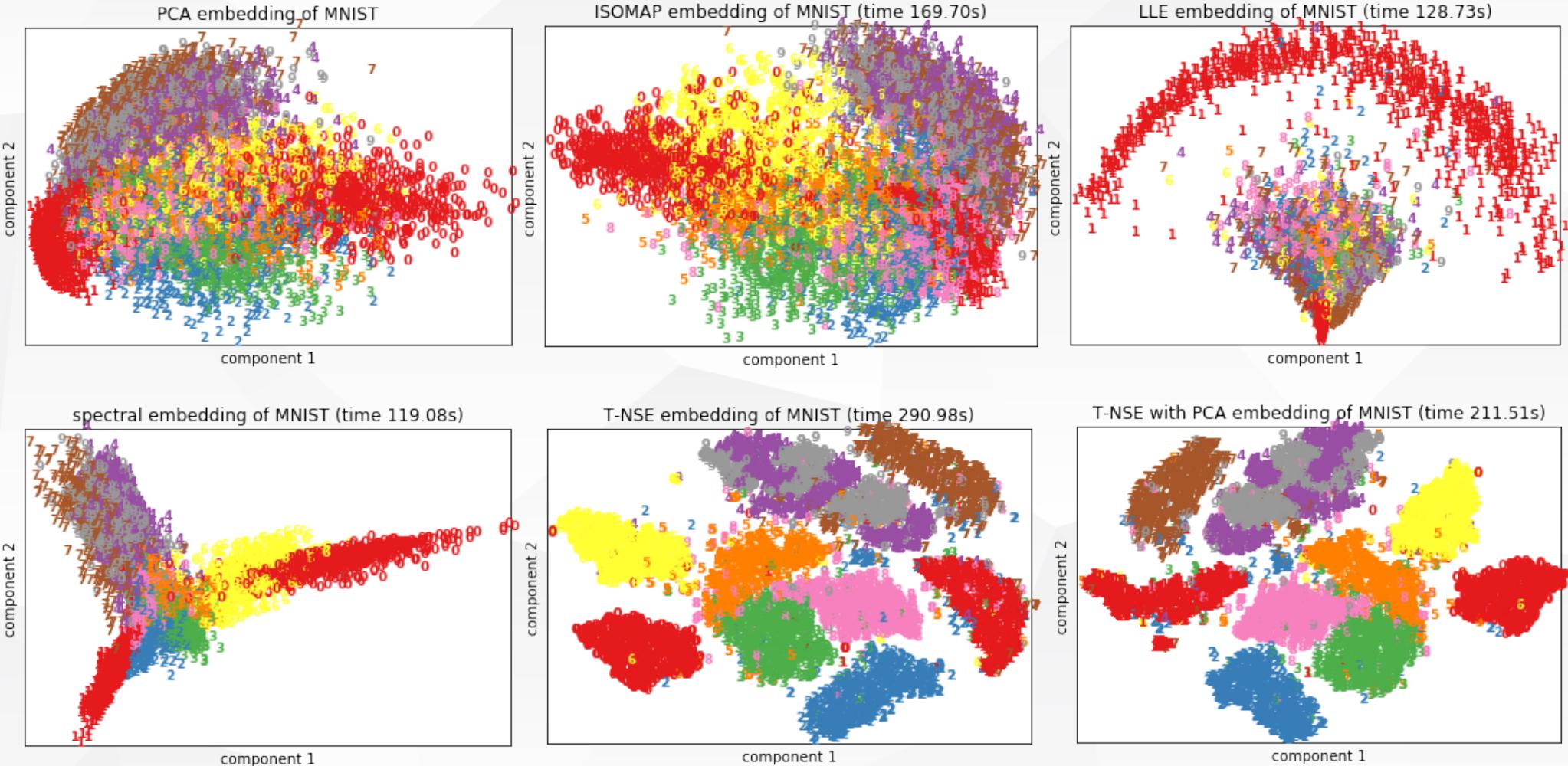


假如权值是5，就转换成5条二元边

➤ 总结

- 降维: 映射原始高维数据到低维空间 : $z = f(x)$
- 算法:
 - 线性 : PCA , NMF , ...
 - 非线性 : KPCA , 深度自编码器、ISOMAP , LLE , LaplaceEigen , Laplacian eigenmap , T-SNE , UMAP、LargeVis , ...

➤ 例：MNIST数据集上的降维结果



➤ 神经网络嵌入：降维和深度学习相结合

- 目标函数与传统降维算法相同，但模型实现采用深度神经网络
 - PCA → 自编码器
- T-SNE → 参数化的T-NSE（参数模型，可得到新的测试样本的低维表示）

➤ 参考文献

- 周志华 , 《机器学习》
- Mika S, Scholkopf B, Smola A. Kernel PCA and de-noising in feature spaces, 1998
- J. B. Tenenbaum, Mapping a manifold of perceptual observations, NIPS, 1998
- Roweis S T, Saul L K. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, 2000
- M. Belkin, P. Niyogi, Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, 2003



The End

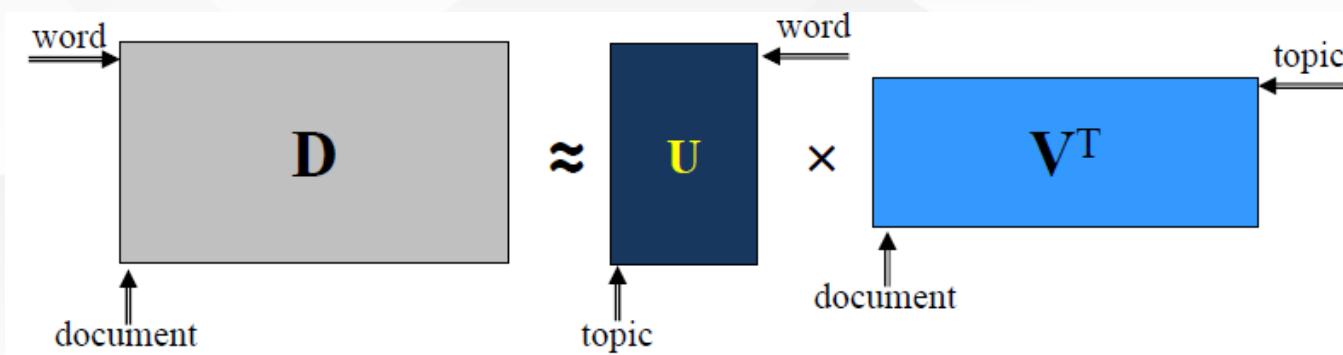
➤ 非负矩阵分解

■许多输入矩阵有非负值

■非负矩阵分解(Non-negative matrix factorization , 简称NMF):

$$X \approx UV^T$$

- X, U 和 V 的元素是非负的
- U 和 V 不再要求是正交的



➤ 非负矩阵分解

■ 目标

$$\min_{\mathbf{U}, \mathbf{V}} \|\mathbf{X} - \mathbf{UV}^T\|^2 \quad \text{s. t. } u_{ij} \geq 0; \ v_{ij} \geq 0$$

$$\|\mathbf{X} - \mathbf{UV}^T\|^2 = \sum_{i=1}^m \sum_{j=1}^d (x_{ij} - u_i v_j^T)^2$$

■ 非凸优化

■ 坐标下降法, 迭代优化:

- 随机初始化 \mathbf{U}
- 固定 \mathbf{U} , 优化 \mathbf{V}
- 固定 \mathbf{V} , 优化 \mathbf{U}
- 重复上述过程直到收敛

➤ NMF算法

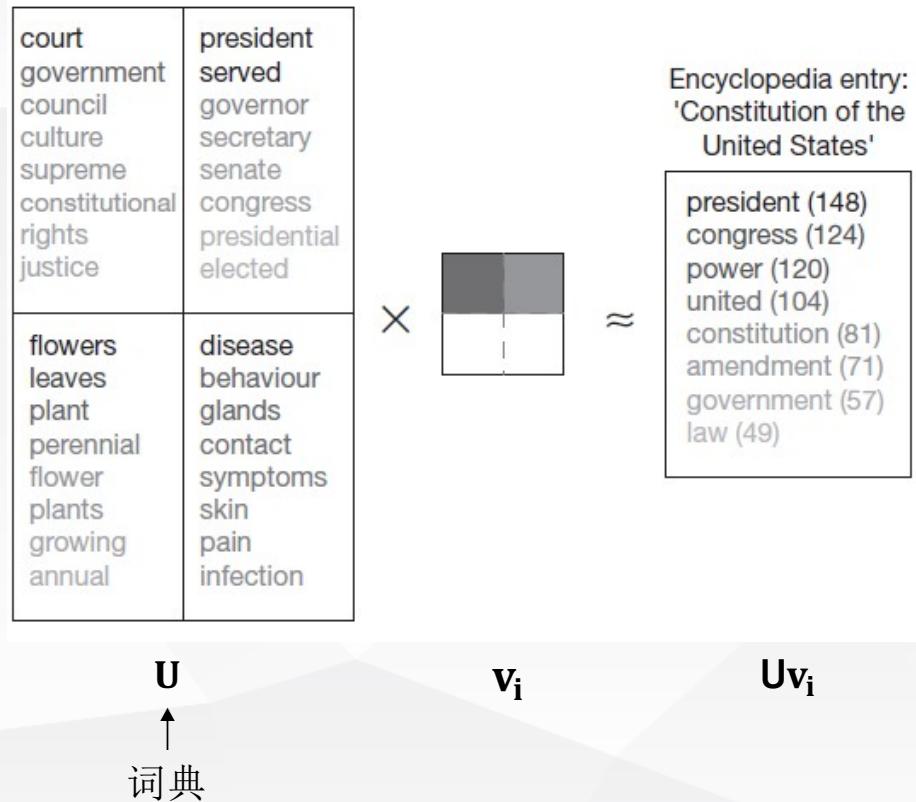
- Input: $X_{d \times m}$, latent dimension K
- Output: U,V
- 1. $U \leftarrow$ random nonnegative values
- 2. repeat
- 3. foreach $v_{ij} \in V$
- 4. $v_{ij} \leftarrow v_{ij} \cdot \frac{(U^T X)_{ij}}{(U^T U V^T)_{ij}}$
- 5. end for
- 6. foreach $u_{ij} \in U$
- 7. $u_{ij} \leftarrow u_{ij} \cdot \frac{(X V)_{ij}}{(X V^T V)_{ij}}$
- 8. end for
- 9. until converge
- 10. return U,V

Theorem 1 The Euclidean distance $\|V - WH\|$ is nonincreasing under the update rules

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}} \quad W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}} \quad (4)$$

The Euclidean distance is invariant under these updates if and only if W and H are at a stationary point of the distance.

➤ NMF应用: 文本分析



Encyclopedia entry:
'Constitution of the
United States'

president (148)
congress (124)
power (120)
united (104)
constitution (81)
amendment (71)
government (57)
law (49)

Figure 4 Non-negative matrix factorization (NMF) discovers semantic features of $m = 30,991$ articles from the Grolier encyclopedia. For each word in a vocabulary of size $n = 15,276$, the number of occurrences was counted in each article and used to form the $15,276 \times 30,991$ matrix V . Each column of V contained the word counts for a particular article, whereas each row of V contained the counts of a particular word in different articles. The matrix was approximately factorized into the form WH using the algorithm described in Fig. 2. Upper left, four of the $r = 200$ semantic features (columns of W). As they are very high-dimensional vectors, each semantic feature is represented by a list of the eight words with highest frequency in that feature. The darkness of the text indicates the relative frequency of each word within a feature. Right, the eight most frequent words and their counts in the encyclopedia entry on the 'Constitution of the United States'. This word count vector was approximated by a superposition that gave high weight to the upper two semantic features, and none to the lower two, as shown by the four shaded squares in the middle indicating the activities of H . The bottom of the figure exhibits the two semantic features containing 'lead' with high frequencies. Judging from the other words in the features, two different meanings of 'lead' are differentiated by NMF.

➤ NMF 总结

■ 要求输入输出矩阵为非负矩阵

- 大多数文本分析任务
- 用户商品矩阵

■ 优化平方损失函数

- 局部最优

■ U 和 V 是可解释的

- U : 学习的词典
- V : 在学习词典下原始数据的表示