软件开发环境国家重点实验室
State Key Laboratory of Software Development Environment

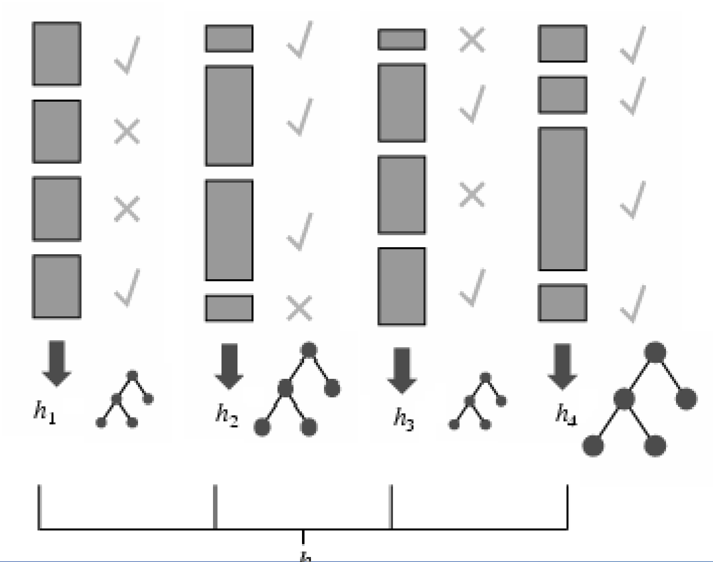
机器学习

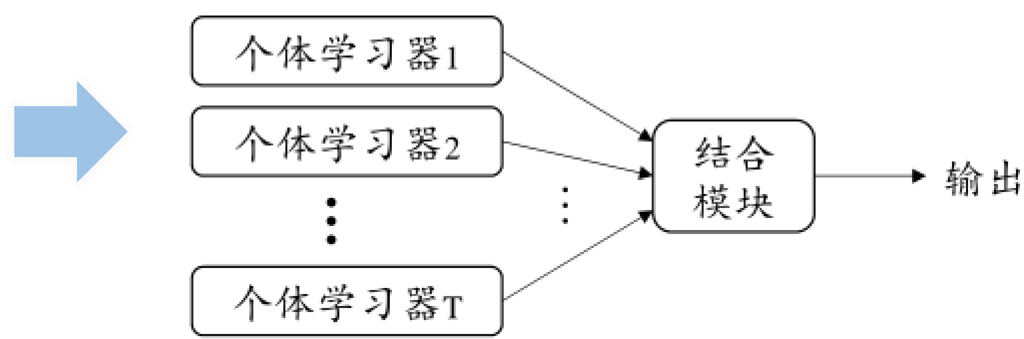
刘祥龙
北京航空航天大学计算机学院
软件开发环境国家重点实验室

2018年11月27日

个体与集成

- No Free Lunch: 没有单一的算法可以保证永远最好
- 集成学习(ensemble learning)通过构建并结合多个学习器来提升性能
- 不同的学习器采用不同的算法、参数、表征（属性、特征、模态等）、训练数据、子问题等





几种典型的集成学习方法

- 有T个弱分类器 y_m，产生强分类器 Y_M

$$Y_M = \frac{1}{M}\sum_{m=1}^{M} y_m$$

Bagging

- 有T个弱分类器，根据各处的权重，产生强分类器

$$Y_M = \frac{1}{M}\sum_{m=1}^{M} \alpha_m y_m$$

Boosting

- 若进一步考虑弱分类器和样本进行自适应

Adaptive Boosting　　Adaboost

Bagging (Bootstrap aggregating)

- 基本步骤
  - 选取T个bootstrap样例 (可重复选取)
  - 在不同的bootstrap样例上训练得到T个不同的分类器（相互独立）
  - 对于新的测试样例，由T个分类器分别预测，并计算平均值（或者多数投票）

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

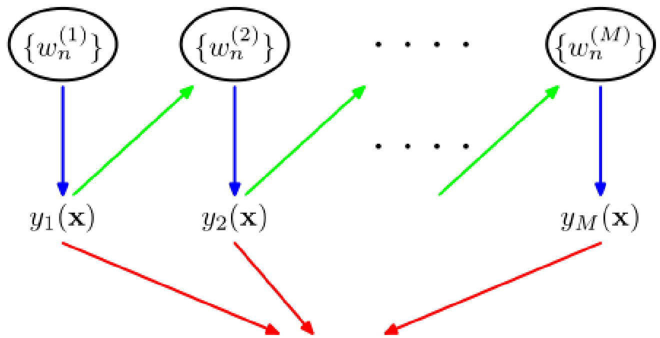
过程:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$
3: **end for**

输出: $H(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\arg\max} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

Boosting

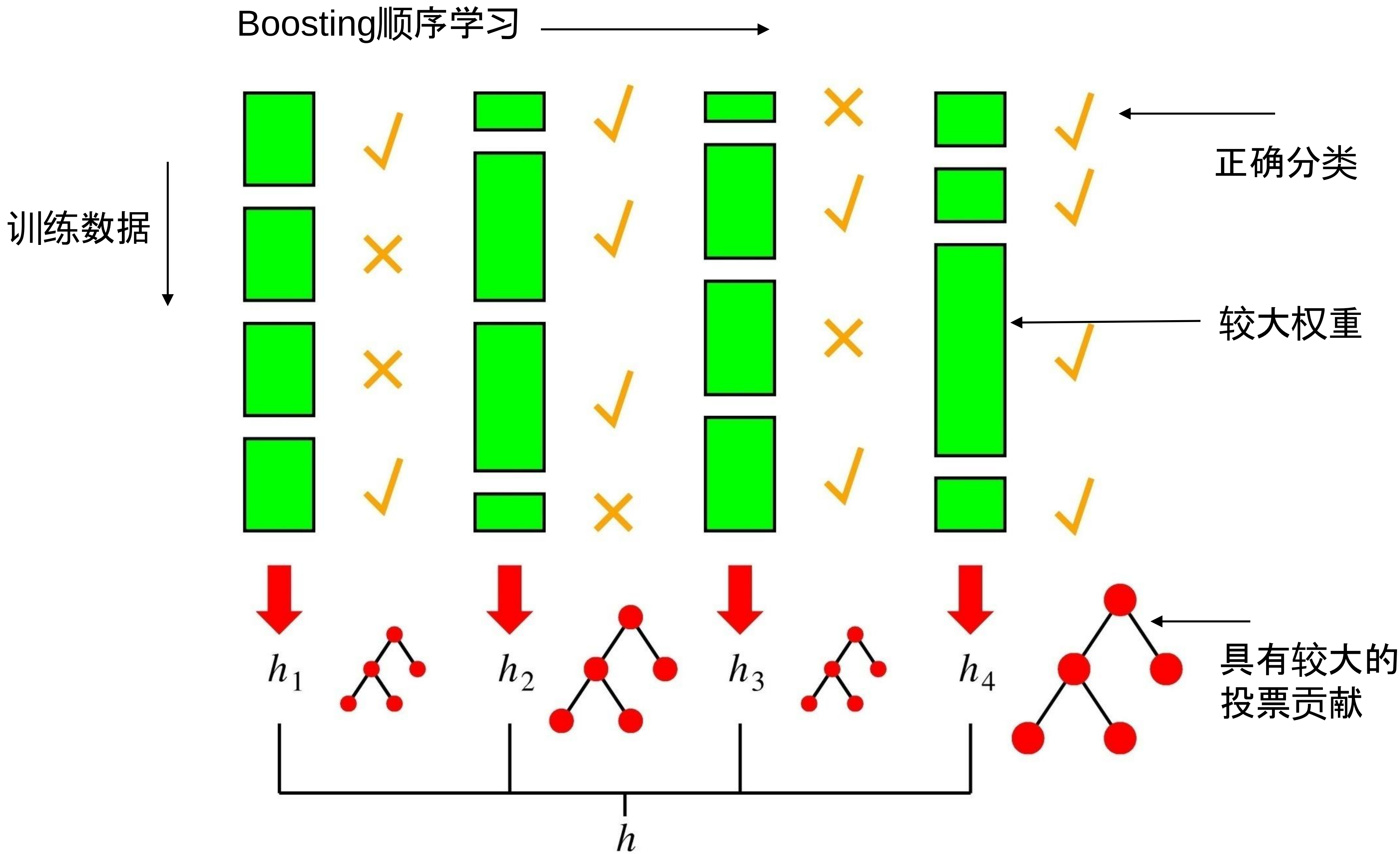
- 基本步骤
  - 顺序训练每个分类器
  - 新的分类器主要集中在上一轮错误分类的样例上
  - 组合所有得到的分类器预测结果



$$Y_M(\mathbf{x}) = \mathrm{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

- 特点
  - 个体学习器存在强依赖关系
  - 训练数据相同，但每次需要调整数据分布
  - 每个分类器是“弱”的，但集成是“强”的

Boosting实例



Boosting – AdaBoost算法

Boosting算法中最著名的代表是AdaBoost

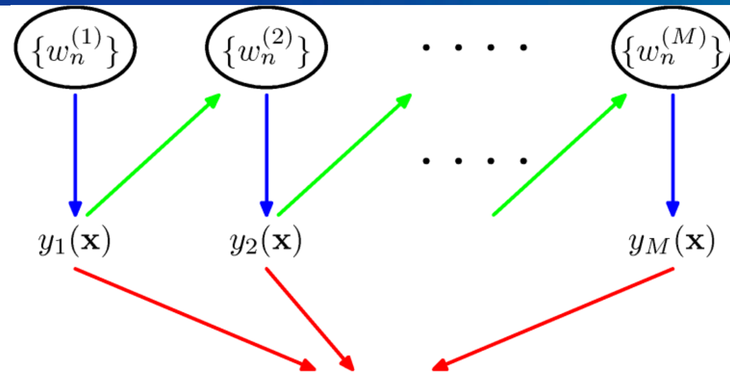
输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程:

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$;
7: $$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases} \\ &= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t} \end{aligned}$$
8: **end for**

输出: $H(\boldsymbol{x}) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \right)$

Boosting – AdaBoost推导



$$Y_M(\mathbf{x}) = \mathrm{sign}\left(\sum_{m}^{M} \alpha_m y_m(\mathbf{x})\right)$$

- 模型：基学习器的线性组合

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

- 损失函数：最小化指数损失函数

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]$$

Boosting – AdaBoost推导

- 优化：
  - 参数α_t：当基分类器h_t基于分布D_t产生后，该基分类器的权重α_t应使得$\alpha_t h_t$最小化指数损失函数

$$
\begin{aligned}
\ell_{\exp}\left(\alpha_t h_t \mid \mathcal{D}_t\right) &= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_t}\left[e^{-f(\boldsymbol{x}) \alpha_t h_t(\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_t}\left[e^{-\alpha_t} \mathbb{I}\left(f(\boldsymbol{x}) = h_t(\boldsymbol{x})\right) + e^{\alpha_t} \mathbb{I}\left(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x})\right)\right] \\
&= e^{-\alpha_t} P_{\boldsymbol{x} \sim \mathcal{D}_t}\left(f(\boldsymbol{x}) = h_t(\boldsymbol{x})\right) + e^{\alpha_t} P_{\boldsymbol{x} \sim \mathcal{D}_t}\left(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x})\right) \\
&= e^{-\alpha_t}\left(1-\epsilon_t\right) + e^{\alpha_t} \epsilon_t
\end{aligned}
\qquad \epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}\left(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x})\right)
$$

  - 对α_t求导为0

$$
\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)
$$

Boosting – AdaBoost推导

- 优化
  - 在获得H_{t-1}之后的样本分布进行调整，使得下一轮的基学习器h_t能纠正H_{t-1}的一些错误，理想的h_t能纠正全部错误

$$h_t(\boldsymbol{x}) = \arg\min_h \ell_{\exp}(H_{t-1} + h \mid \mathcal{D})$$

$$= \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1 - f \right.\right.$$

$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} f(\boldsymbol{x})h(\boldsymbol{x})\right]$$

$$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})\, e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]}$$

$$= \frac{\mathcal{D}(\boldsymbol{x})\, e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]}$$

$$= \mathcal{D}_t(\boldsymbol{x}) \cdot e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})} \frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]}$$

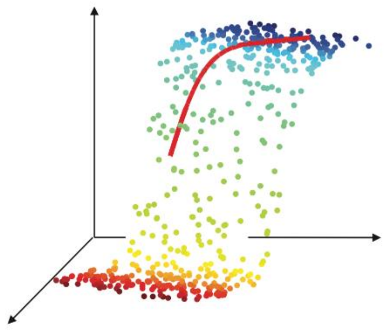
$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})\right],$$

$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right].$$

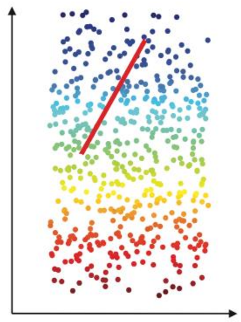
分类器

低维嵌入

- 缓解维数灾难的一个重要途径是降维(dimension reduction)
  - 即通过某种数学变换，将原始高维属性空间转变为一个低维“子空间”（subspace)，在这个子空间中样本密度大幅度提高，距离计算也变得更为容易。
- 为什么能进行降维?
  - 数据样本虽然是高维的，但与学习任务密切相关的也许仅是某个低维分布， 即高维空间中的一个低维“嵌入”(embedding)，因而可以对数据进行有效的降维。



(a) 三维空间中观察到的样本点



(b) 二维空间中的曲面

线性降维方法

- 对原始高维空间进行线性变换。给定d维空间中的样本 $\mathbf{X}=(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m)\in\mathbb{R}^{d\times m}$ ，变换之后得到 $d'\leq d$ 维空间中的样本
$$\mathbf{Z}=\mathbf{W}^{\mathrm{T}}\mathbf{X},$$
其中$\mathbf{W}\in\mathbb{R}^{d\times d'}$是变换矩阵，$\mathbf{Z}\in\mathbb{R}^{d'\times m}$是样本在新空间中的表达。

- 变换矩阵 W 可视为 d'个d 维属性向量。换言之，z_i是原属性向量 $\boldsymbol{x}_i$ 在新坐标系 $\{\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots,\boldsymbol{w}_{d'}\}$ 中的坐标向量。若 $\boldsymbol{w}_i$ 与 $\boldsymbol{w}_j(i\neq j)$正交，则新坐标系是一个正交坐标系，此时 W为正交变换。
- 显然，新空间中的属性是原空间中的属性的线性组合。

主成分分析

- 降维：对于正交属性空间中的样本点，如何用一个超平面对所有样本进行恰当的表达?

- 容易想到，若存在这样的超平面，那么它大概应具有这样的性质：
  - 使得降维后最大程度保持原始的数据特性：方差最大
  - 使得降维后数据的误差尽可能小：均方误差最小

- 能分别得到主成分分析的两种等价推导。

最大化方差

- 基本思想：使用较少的数据维度保留数据特性（方差）
- 将D维数据集 $\{\mathbf{x}_n\}, n = 1, 2, ..., N$ 降为 $M < D$，不失一般性，先考虑 $M = 1$，投影为 $\mathbf{u}_1$，$\mathbf{u}_1^T\mathbf{u}_1 = 1$
- 模型：
  - 每个数据点 $\mathbf{x}_n$ 在新空间中表示为标量 $\mathbf{u}_1^T\mathbf{x}_n$
  - 样本均值在新空间中表示为 $\mathbf{u}_1^T\bar{\mathbf{x}}$，其中 $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$
- 投影后样本方差表示为

$$\frac{1}{N}\sum_{n=1}^{N}\{\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}}\}^2 = \boxed{\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1}$$ **最大**

- 其中原样本方差 $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$

最大化方差

- 基本思想：使用较少的数据维度保留原数据特性

- 优化目标：　max　$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1, \quad s.t. \quad \mathbf{u}_1^T \mathbf{u}_1 = 1$

- 求解：利用拉格朗日乘子法 $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1)$
  - 对 $\mathbf{u}_1$ 求导置零得到

$$\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

u_1是S的特征向量

  - 进一步得到

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1$$

u_1是S最大特征值对应的特征向量时
方差取到极大值，称u_1为第一主成分

最小化误差

- 优化目标：最小化失真度

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \sum_{i=M+1}^{D} \{(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_i\} \mathbf{u}_i$$

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i)^2 = \sum_{i=M+1}^{D} \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i$$

- 优化：拉格朗日乘子法

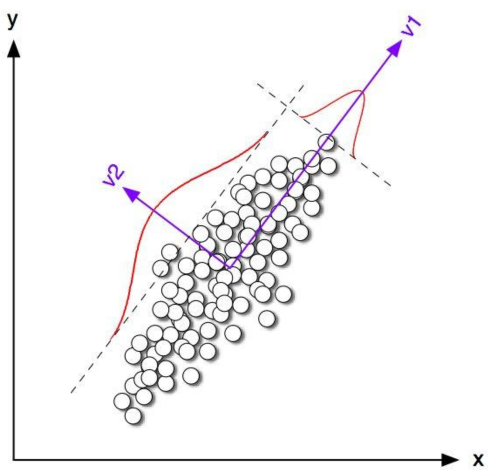
$$\tilde{J} = \sum_{i=M+1}^{D} \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i + \sum_{i=M+1}^{D} \lambda_i (1 - \mathbf{u}_i^T \mathbf{u}_i)$$

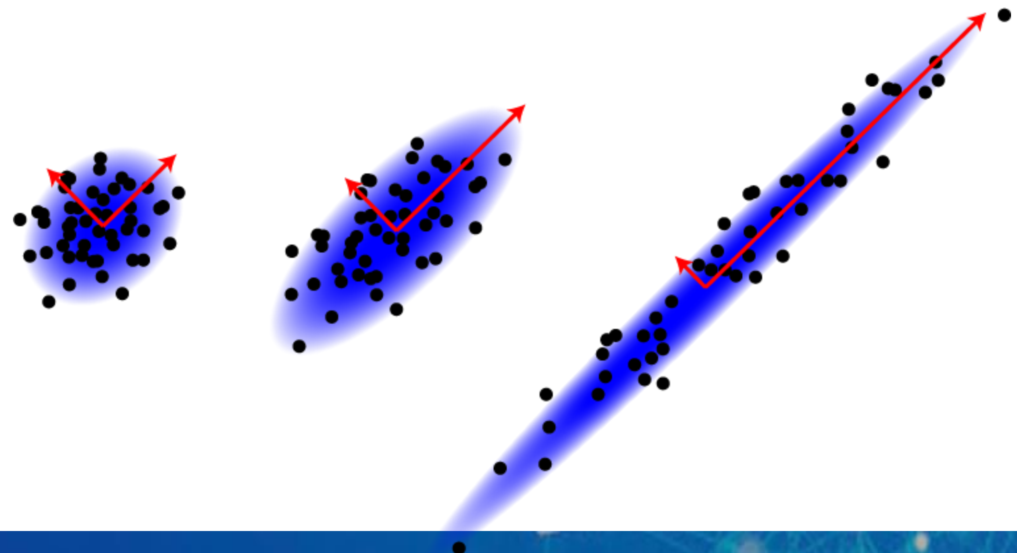
- 求导得到 $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ **J最小时取D-M个最小的特征值**

- 对应失真度为 $J = \sum_{i=M+1}^{D} \lambda_i$ **主子空间对应M个最大特征值**

主成分分析-算法

- 计算步骤
- ①计算给定样本 $\{\mathbf{x}_n\}, n = 1, 2, ..., N$ 的均值 $\bar{\mathbf{x}}$ 和协方差矩阵S;
- ②计算S的特征向量与特征值, $X = U\Lambda U^T$;
- ③将特征值从大到小排列，前M个特征值 $\lambda_1, ..., \lambda_M$ 所对应的特征向量 $\mathbf{u}_1, ..., \mathbf{u}_M$ 构成投影矩阵。





主成分分析-应用

特征脸(Eigenfaces)#1~#8

核主成分分析（Kernel PCA）

- 将主成分分析的线性假设一般化使之适应非线性数据
- 传统PCA：D维样本 $\{\mathbf{x}_n\}, n = 1, 2, ..., N$ ，$\sum_n \mathbf{x}_n = \mathbf{0}$

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \quad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} \quad \mathbf{u}_i^T \mathbf{u}_i = 1$$

- 核PCA：非线性映射 $\phi(\mathbf{x})$，$\mathbf{x}_n \longmapsto \phi(\mathbf{x}_n)$，$\Sigma_n \phi(\mathbf{x}_n) = 0$

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \qquad \mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^{\mathrm{T}}$$

$$\longrightarrow \frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^T \mathbf{v}_i\} = \lambda_i \mathbf{v}_i$$

核主成分分析

- 新的数据空间下 $\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\boldsymbol{\phi}(\mathbf{x}_n)$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T\sum_{m=1}^{N}a_{im}\phi(\mathbf{x}_m) = \lambda_i\sum_{n=1}^{N}a_{in}\phi(\mathbf{x}_n)$$

$$\boxed{k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m)}$$

$$\longrightarrow \frac{1}{N}\sum_{n=1}^{N}k(\mathbf{x}_l, \mathbf{x}_n)\sum_{m=1}^{N}a_{im}k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i\sum_{n=1}^{N}a_{in}k(\mathbf{x}_l, \mathbf{x}_n)$$

$$\longrightarrow \mathbf{K}^2\mathbf{a}_i = \lambda_i N\mathbf{K}\mathbf{a}_i$$

$$\longrightarrow \mathbf{K}\mathbf{a}_i = \lambda_i N\mathbf{a}_i$$

流形学习

流形学习(manifold learning)

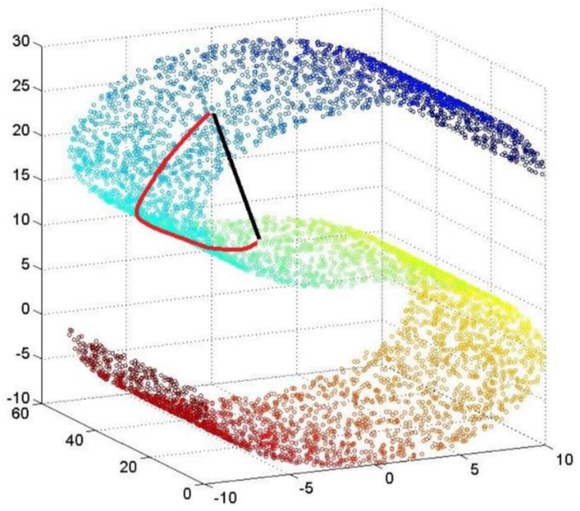
- “流形”是在局部与欧氏空间同胚的空间，换言之，它在局部具有欧氏空间的性质，能用欧氏距离来进行距离计算。

- 一类借鉴了拓扑流形概念的降维方法：若低维流形嵌入到高维空间中，则数据样本在高维空间的分布虽然看上去非常复杂，但在局部上仍具有欧氏空间的性质，因此，可以容易地在局部建立降维映射关系，然后再设法将局部映射关系推广到全局。

- 当维数被降至二维或三维时，能对数据进行可视化展示，因此流形学习也可被用于可视化。

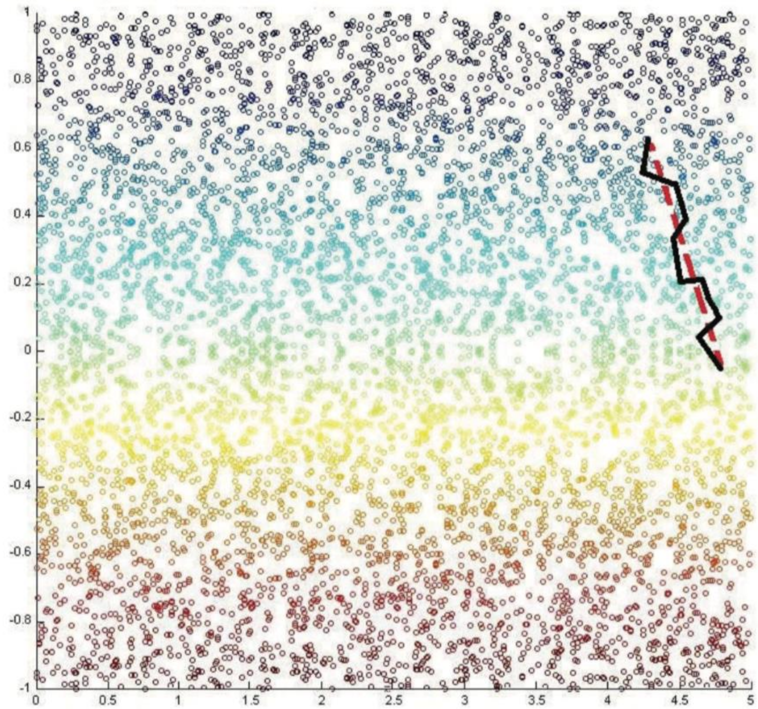
等距映射（Isometric Mapping, Isomap）

- 低维流形嵌入到高维空间之后，直接在高维空间中计算直线距离具有误导性，因为高维空间中的直线距离在低维嵌入流形上不可达
- 低维嵌入流形上两点间的本真距离是“测地线”(geodesic)距离。



等距映射（Isometric Mapping, Isomap）

- 测地线距离的计算
  - ❑ 利用流形在局部上与欧氏空间同胚这个性质，对每个点基于欧氏距离找出其近邻点，然后就能建立一个近邻连接图，图种近邻点之间存在连接，而非近邻点之间不存在连接，于是，计算两点之间测地线距离的问题，就转变为计算近邻连接图上两点之间的最短路径问题。
  - ❑ 最短路径的计算可通过Dijkstra算法或Floyd算法实现。得到距离后可通过多维缩放方法获得样本点在低维空间中的坐标。



(b) 测地线距离与近邻距离

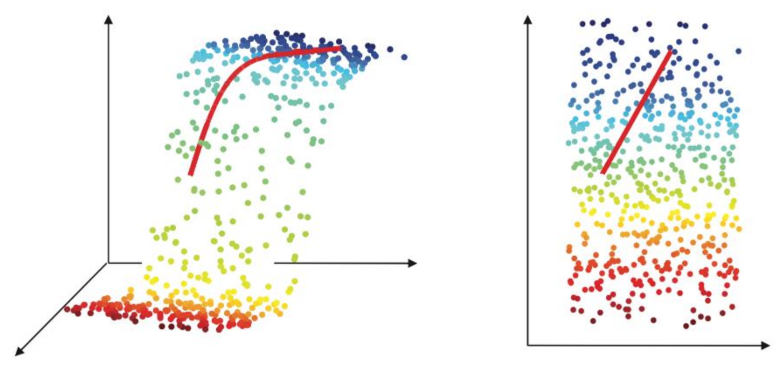
多维缩放

- 若要求原始空间中样本之间的距离在低维空间中得以保持，即得到“多维缩放”(Multiple Dimensional Scaling, MDS)

  - 假定有m个样本，在原始空间中的距离矩阵为$\mathbf{D} \in \mathbb{R}^{m\times m}$，其第i行j列的元素$dist_{ij}$为样本$\boldsymbol{x}_i$到$\boldsymbol{x}_j$的距离。
  - 目标是获得样本在d'维空间中的欧氏距离等于原始空间中的距离，即$||\boldsymbol{z}_i - \boldsymbol{z}_j|| = dist_{ij}$.
  - 令$\mathbf{B} = \mathbf{Z}^{\mathrm{T}}\mathbf{Z} \in \mathbb{R}^{m\times m}$，其中B为降维后的内积矩阵，$b_{ij} = \boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{z}_j$，有

$$\begin{aligned} dist_{ij}^2 &= ||\boldsymbol{z}_i||^2 + ||\boldsymbol{z}_j||^2 - 2\boldsymbol{z}_i^T\boldsymbol{z}_j \\ &= b_{ii} + b_{jj} - 2b_{ij}. \end{aligned}$$

  - 可通过降维前的距离矩阵D获得



(a) 三维空间中观察到的样本点　　(b) 二维空间中的曲面

图 10.2　低维嵌入示意图

多维缩放

- 为便于讨论，令降维后的样本 Z 被中心化，即 $\sum_{i=1}^{m} \boldsymbol{z}_i = \mathbf{0}$ 。显然，矩阵 B 的行与列之和均为零，即

$$\sum_{i=1}^{m} b_{ij} = \sum_{j=1}^{m} b_{ij} = 0.$$

由

$$\begin{aligned} dist_{ij}^2 &= ||\boldsymbol{z}_i||^2 + ||\boldsymbol{z}_j||^2 - 2\boldsymbol{z}_i^T \boldsymbol{z}_j \\ &= b_{ii} + b_{jj} - 2b_{ij}. \end{aligned}$$

易知

$$\sum_{i=1}^{m} dist_{ij}^2 = \mathrm{tr}(\mathbf{B}) + m b_{jj}, \quad \sum_{j=1}^{m} dist_{ij}^2 = \mathrm{tr}(\mathbf{B}) + m b_{ii}, \quad \sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2 = 2m\,\mathrm{tr}(\mathbf{B}),$$

多维缩放

$$\sum_{i=1}^{m} dist_{ij}^2 = \mathrm{tr}(\mathbf{B}) + m b_{jj}, \quad \sum_{j=1}^{m} dist_{ij}^2 = \mathrm{tr}(\mathbf{B}) + m b_{ii}, \quad \sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2 = 2m\ \mathrm{tr}(\mathbf{B}),$$

记:

$$dist_{i\cdot}^2 = \frac{1}{m}\sum_{j=1}^{m} dist_{ij}^2$$

$$tr(\mathbf{B}) = \frac{1}{2m}\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2.$$

$$dist_{\cdot j}^2 = \frac{1}{m}\sum_{i=1}^{m} dist_{ij}^2$$

$$dist_{\cdot\cdot}^2 = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} dist_{ij}^2$$

则:

$$b_{ii} = \frac{1}{m}\left(m \cdot dist_{i\cdot}^2 - \frac{m}{2} dist_{\cdot\cdot}^2\right),$$

$$b_{jj} = \frac{1}{m}\left(m \cdot dist_{\cdot j}^2 - \frac{m}{2} dist_{\cdot\cdot}^2\right).$$

$$tr(\mathbf{B}) = \frac{m}{2} dist_{\cdot\cdot}^2$$

多维缩放

由：

$$b_{ii} = \frac{1}{m}(m \cdot dist_{i\cdot}^2 - \frac{m}{2} dist_{\cdot\cdot}^2),$$

$$b_{jj} = \frac{1}{m}(m \cdot dist_{\cdot j}^2 - \frac{m}{2} dist_{\cdot\cdot}^2).$$

则：

$$\begin{aligned} dist_{ij}^2 &= ||\boldsymbol{z}_i||^2 + ||\boldsymbol{z}_j||^2 - 2\boldsymbol{z}_i^T \boldsymbol{z}_j \\ &= b_{ii} + b_{jj} - 2b_{ij}. \end{aligned} \Rightarrow b_{ij} = \frac{1}{2}(b_{ii} + b_{jj} - dist_{ij}^2)$$

$$b_{ij} = \frac{1}{2}(dist_{i\cdot}^2 + dist_{\cdot j}^2 - dist_{\cdot\cdot}^2 - dist_{ij}^2)$$

多维缩放

- 对矩阵 $\mathbf{B}$ 做特征值分解 $\mathbf{B}=\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathrm{T}}$，其中

$$\mathbf{\Lambda}=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_d)$$

为特征值构成的对角矩阵

- 在现实应用中为了有效降维，往往仅需降维后的距离与原始空间中的距离尽可能接近，而不必严格相等。此时可取 $d'\ll d$ 个最大特征值构成对角矩阵，令 $\tilde{\mathrm{V}}$ 表示相应的特征向量矩阵，则 z 可表达为

$$\mathbf{Z}=\tilde{\mathbf{\Lambda}}^{1/2}\tilde{\mathbf{V}}^{\mathrm{T}}\in\mathbb{R}^{d'\times m}.$$

$$\tilde{\mathbf{\Lambda}}=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_{d'})$$

等距映射

●计算步骤

①构造临近关系图

对每一个点，将它与指定半径邻域内所有点相连(或与指定个数最近邻相连)

②计算最短路径

计算临近关系图所有点对之间的最短路径，得到距离矩阵

③多尺度分析

将高维空间中的数据点投影到低维空间，使投影前后的距离矩阵相似度最大

等距映射（Isometric Mapping, Isomap）

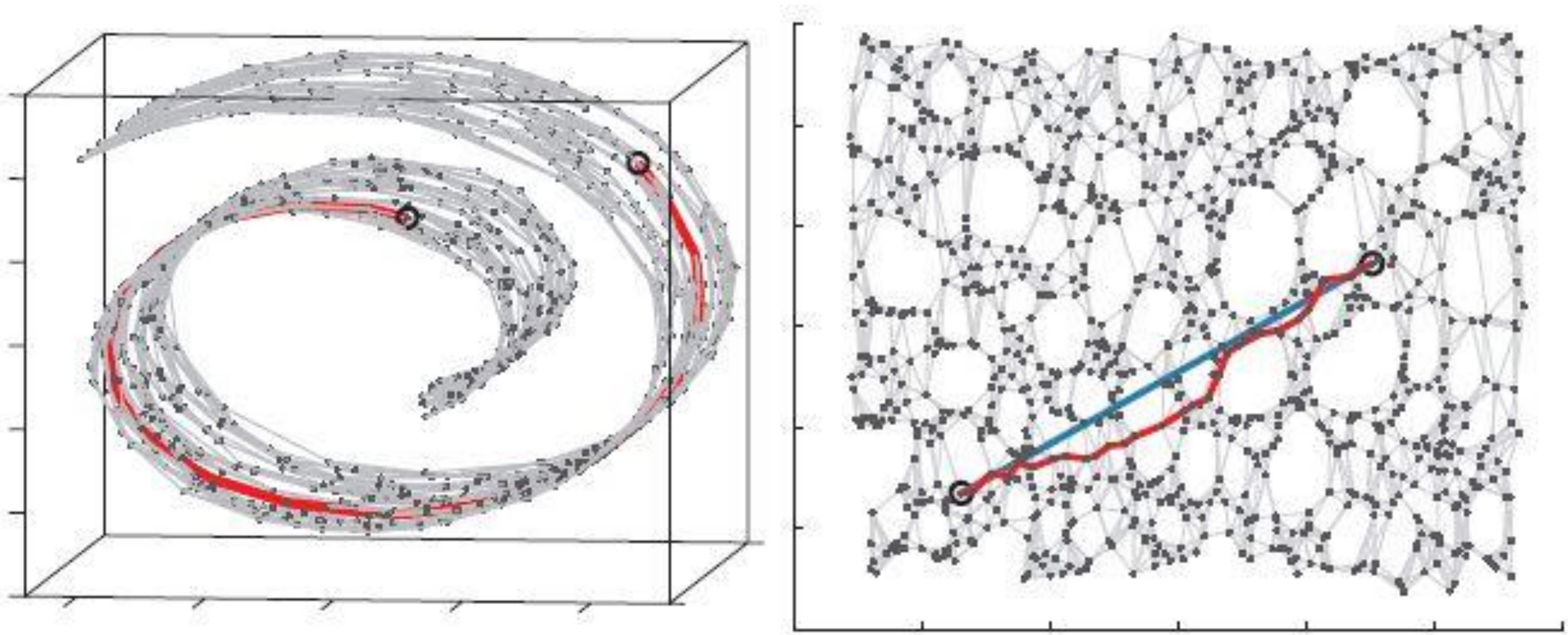
输入： 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
近邻参数 k;
低维空间维数 d'.

过程：

for $i = 1, 2, \ldots, m$ **do**
    确定 $\boldsymbol{x}_i$ 的 k 近邻;
    $\boldsymbol{x}_i$ 与 k 近邻点之间的距离设置为欧氏距离, 与其他点的距离设置为无穷大;
end for
调用最短路径算法计算任意两样本点之间的距离 $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$;
将 $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 作为 MDS 算法的输入;
return MDS 算法的输出

输出： 样本集 D 在低维空间的投影 $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m\}$.

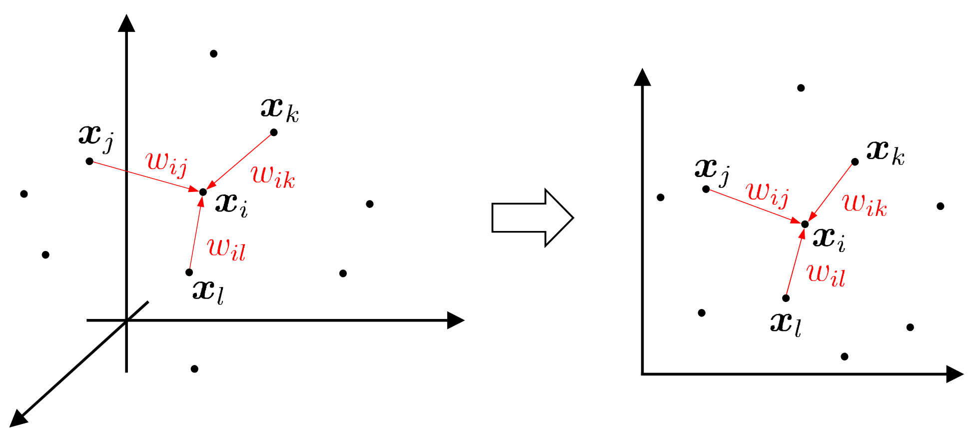
等距映射



K=7, *N*=1000

局部线性嵌入 (Locally Linear Embedding，LLE)

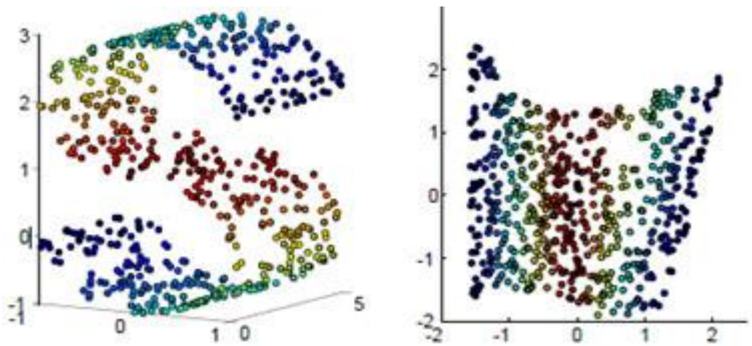
- 局部线性嵌入试图保持邻域内的线性关系，并使得该线性关系在降维后的空间中继续保持。



$$\boldsymbol{x}_i = w_{ij}\boldsymbol{x}_j + w_{ik}\boldsymbol{x}_k + w_{il}\boldsymbol{x}_l$$

局部线性嵌入 (Locally Linear Embedding，LLE)

- Local Linear Embedding (LLE)
- 保持数据点的原有流形结构



Nonlinear Dimensionality Reduction by Locally Linear Embedding

Sam T. Roweis[1] and Lawrence K. Saul[2]

SCIENCE VOL 290 22 DECEMBER 2000

- 前提假设：采样数据所在的低维流形在局部是线性的，每个采样点可以用它的近邻点线性表示。
- 学习目标：在低维空间中保持每个邻域中的权值不变，即假设嵌入映射在局部是线性的条件下，最小化重构误差。

局部线性嵌入 (Locally Linear Embedding，LLE)

- 优化目标：LLE先为每个样本 $\boldsymbol{x}_i$找到其近邻下标集合 Q_i，然后计算出基于Q_i 的中的样本点对$\boldsymbol{x}_i$ 进行线性重构的系数 $\boldsymbol{w}_i$

$$\min_{\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots,\boldsymbol{w}_m} \sum_{i=1}^{m} \left\| \boldsymbol{x}_i - \sum_{j\in Q_i} w_{ij}\boldsymbol{x}_j \right\|_2^2$$

$$\text{s.t.} \sum_{j\in Q_i} w_{ij} = 1,$$

- 其中$\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 均为已知，令 $C_{jk} = (\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}}(\boldsymbol{x}_i - \boldsymbol{x}_k)$，$w_{ij}$ 有闭式解

$$w_{ij} = \frac{\sum_{k\in Q_i} C_{jk}^{-1}}{\sum_{l,s\in Q_i} C_{ls}^{-1}}.$$

局部线性嵌入 (Locally Linear Embedding，LLE)

- 优化目标：LLE在低维空间中保持 $\boldsymbol{w}_i$不变，于是 $\boldsymbol{x}_i$ 对应的低维空间坐标 z_i 可通过下式求解：

$$\min_{\boldsymbol{z}_i} \quad \sum_{i=1}^{m} \left\| \boldsymbol{z}_i - \sum_{j \in Q_i} w_{ij} \boldsymbol{z}_j \right\|_2^2$$

- 令 $\mathbf{Z} = (\mathrm{z}_1, \mathrm{z}_2, \ldots, \ \mathrm{z}_m) \in \mathbb{R}^{d' \times m}, (\mathbf{W})_{ij} = w_{ij},$

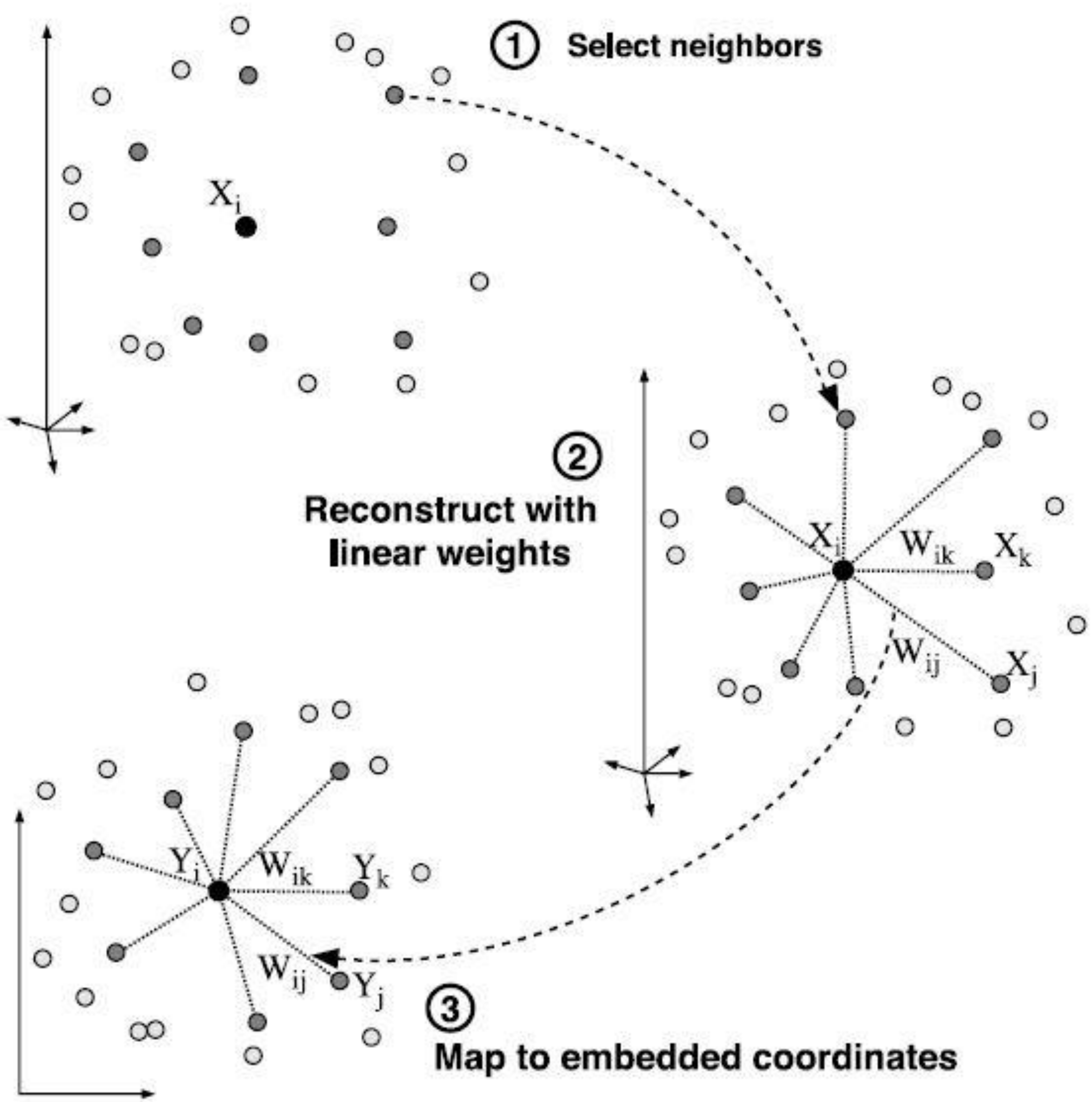
$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^{\mathrm{T}} (\mathbf{I} - \mathbf{W}),$$

  - 则优化式可重写为右式，并通过特征值分解求解。

$$\min_{\mathbf{Z}} \ \mathrm{tr}(\mathbf{Z}\mathbf{M}\mathbf{Z}^{\mathrm{T}})$$
$$\text{s.t. } \mathbf{Z}\mathbf{Z}^{\mathrm{T}} = \mathbf{I}.$$

局部线性嵌入

- 计算步骤



① Select neighbors
X_i
② Reconstruct with linear weights
X_i W_{ik} X_k
W_{ij} X_j
Y_i W_{ik} Y_k
W_{ij} Y_j
③ Map to embedded coordinates

局部线性嵌入 (Locally Linear Embedding，LLE)

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
近邻参数 k;
低维空间维数 d'.

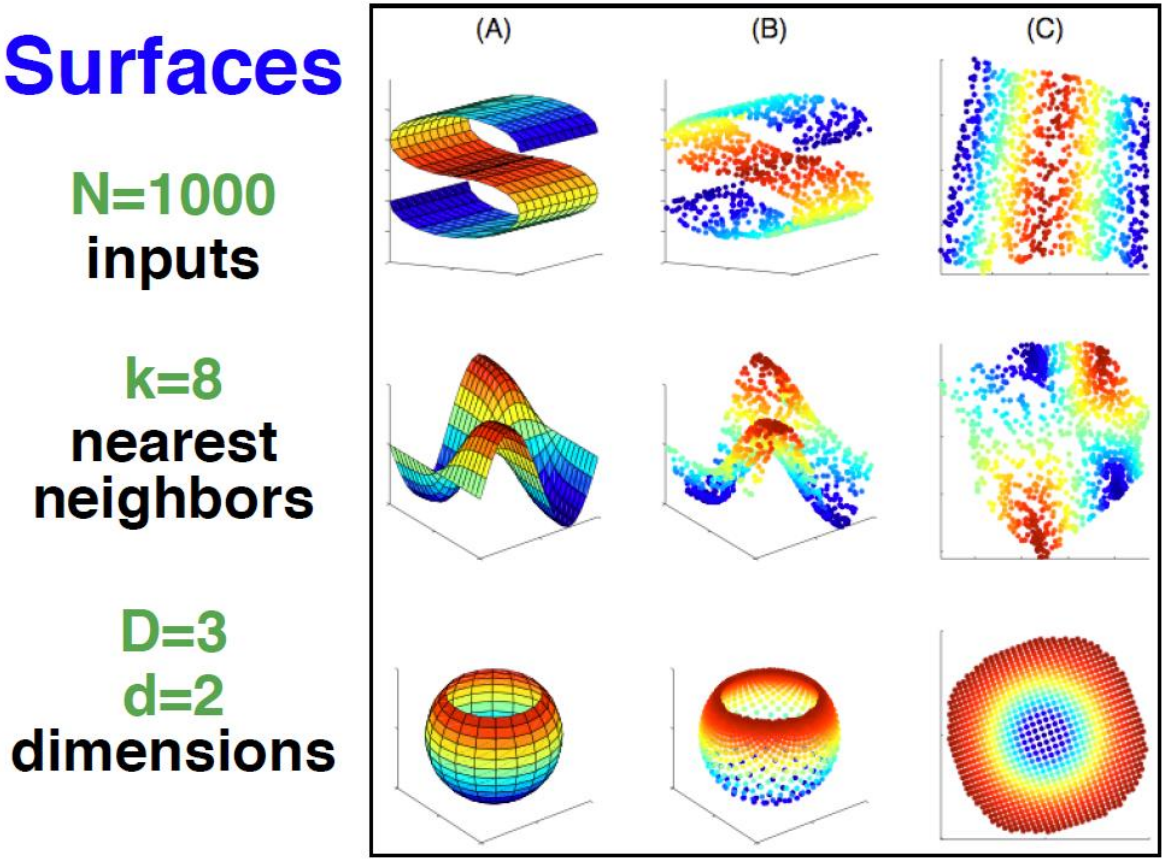
过程:

```
for i = 1, 2, ..., m do
    确定 x_i 的 k 近邻;
    从式(10.27)求得 w_ij, j ∈ Q_i;
    对于 j ∉ Q_i, 令 w_ij = 0;
end for
从式(10.30)得到 M;
对 M 进行特征值分解;
return M 的最小 d' 个特征值对应的特征向量
```

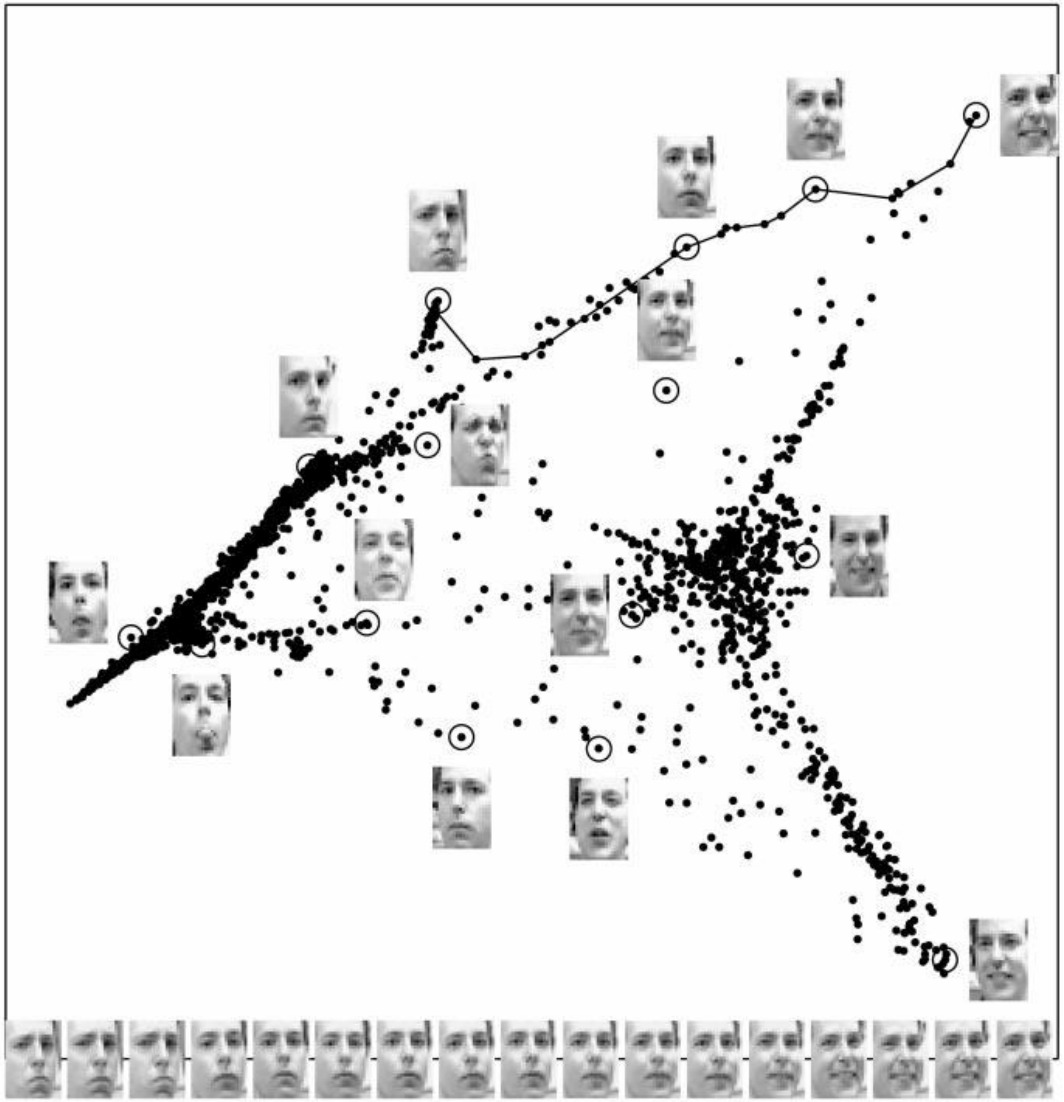
输出: 样本集 D 在低维空间的投影 $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m\}$.

局部线性嵌入 (Locally Linear Embedding，LLE)

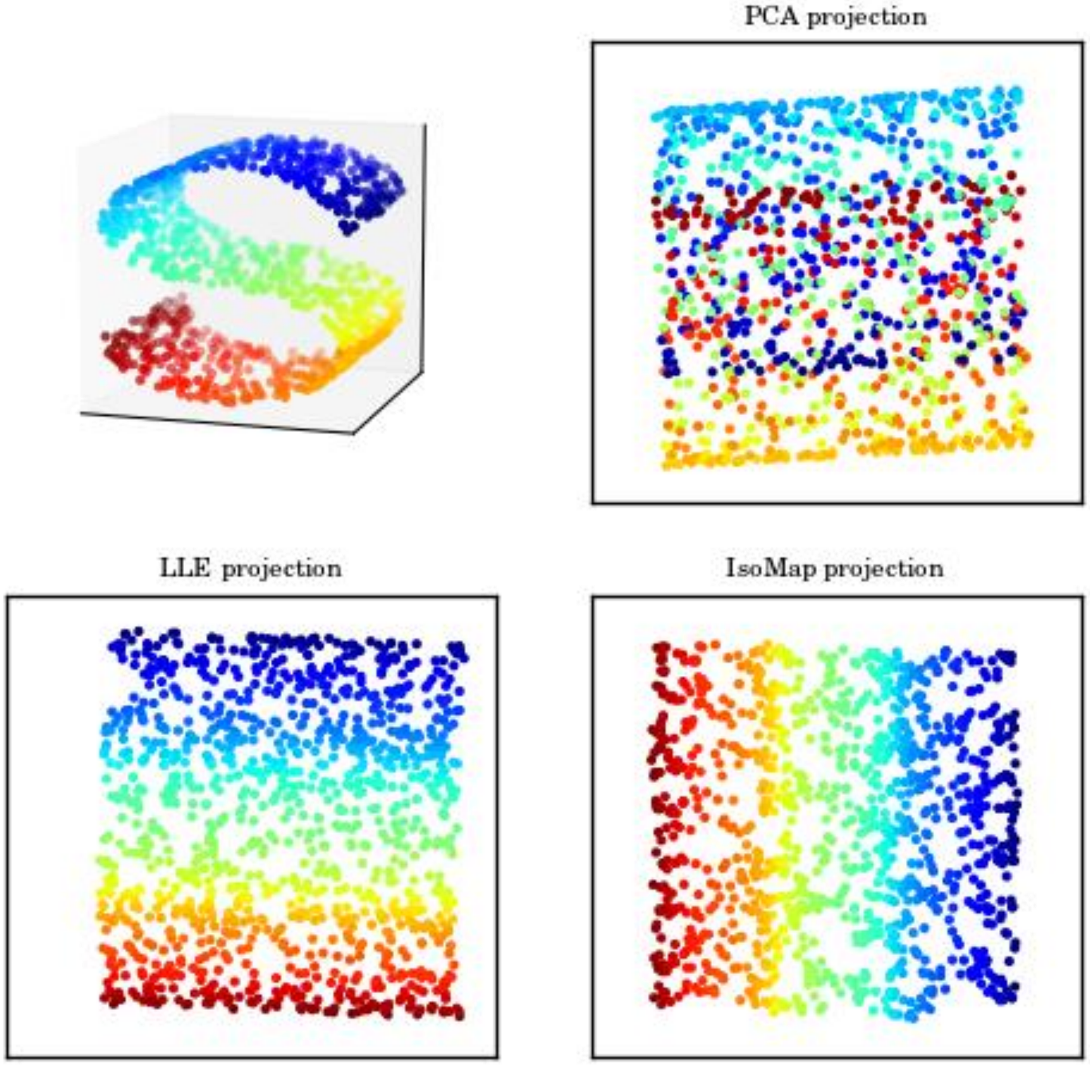
- Local Linear Embedding (LLE)



局部线性嵌入 (Locally Linear Embedding，LLE)



降维方法对比



主成分分析-问题

- 利用PCA处理高维数据
  - 在实际应用中，样本维数可能很高，远大于样本的个数
  - 在人脸识别中，1000张人脸图像，每张图像100×100像素
  - D维空间，N个样本点，X是N×D维的数据矩阵

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T \longrightarrow \mathbf{S} = N^{-1}\mathbf{X}^T\mathbf{X}$$

S维数? $D \times D$**维** 10000×10000

主成分分析-问题

- 利用PCA处理高维数据
  - 在实际应用中，样本维数可能很高，远大于样本的个数
  - 在人脸识别中，1000张人脸图像，每张图像100×100像素
  - 对 $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ 求的特征值 λ_i 和特征向量 $\mathbf{v}_i$

$$\boxed{\frac{1}{N}\mathbf{X}\mathbf{X}^T}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

***D×D*维**

  - 如何从 $\mathbf{v}_i$ 到 $\mathbf{u}$

$$\longrightarrow \quad \frac{1}{N}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{v}_i) = \lambda_i \boxed{(\mathbf{X}^T\mathbf{v}_i)}$$

S的特征向量

$$\mathbf{u}_i \propto \mathbf{X}^T\mathbf{v}_i \qquad \|\mathbf{u}_i\| = 1$$

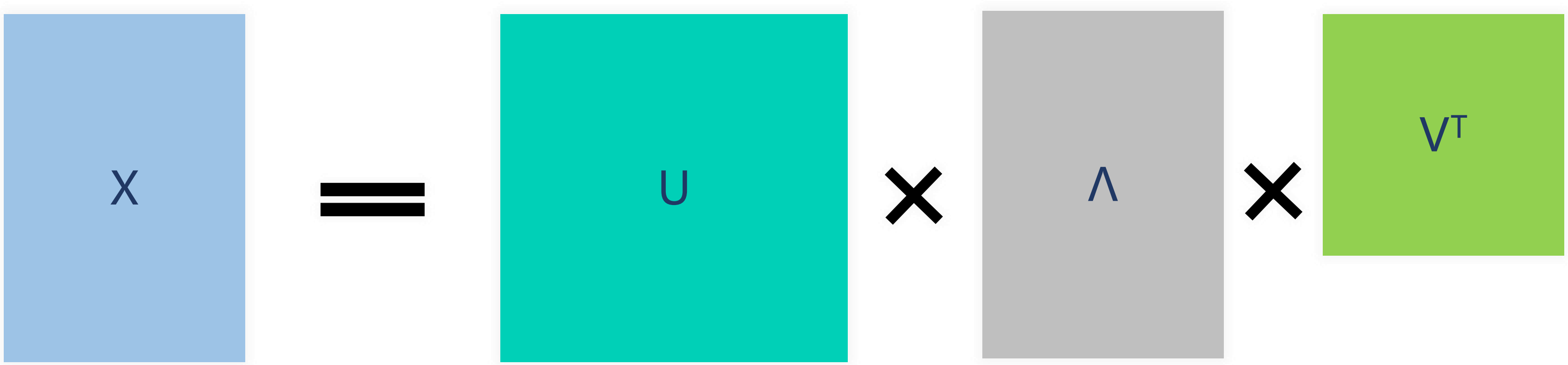
$$\longrightarrow \quad \mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}}\mathbf{X}^T\mathbf{v}_i$$

奇异值分解(Singular Value Decomposition, SVD)

奇异值分解 (Singular Value Decomposition, SVD)

- 矩阵$X \in R^{n\times m}$存在以下分解

$$X = U \Lambda V^{T} = \sum_{k=1}^{r} u_k \lambda_k v_k^T$$



$U^T U = I$ $\quad$ $V^T V = I$

与PCA的关系

- 而特征分解只能适用于特定类型的方阵，故奇异值分解的适用范围更广
- 但二者存在关联：
  - 对任意矩阵X $\in R^{n\times m}$

$$X = U\Lambda V^{T} = \sum_{k=1}^{r} u_k \lambda_k v_k^{T}$$

  - $X^TX \in R^{m\times m}$

$$X^TX = (U\Lambda V^T)^T(U\Lambda V^T) = V\Lambda U^TU\Lambda V^T = V\Lambda^2V^T$$

  - $XX^T \in R^{n\times n}$

$$XX^T = (U\Lambda V^T)(U\Lambda V^T)^T = U\Lambda V^TV\Lambda U^T = U\Lambda^2U^T$$

  - $\mathbf{X^TU = (U\Lambda V^T)^TU = V\Lambda}$

低秩近似（Low-rank Approximation）

- 矩阵低秩近似 $$\tilde{A} = \min_{A:rank(A)=k} \left\| A - X \right\|_F$$

$$\tilde{A} = U\ \mathrm{diag}(\lambda_1, \ldots, \lambda_k, \underbrace{0, \ldots, 0}_{\text{设置最小的 m-k 奇异值为0}}) V^T$$

$$\underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{V^T}$$

$K=2$

$$\tilde{A} = \sum_{i=1}^{k} \lambda_i u_i v_i^T$$

奇异值分解应用

- 文档检索：原始矩阵A
  - 术语i和j有多相似?
  - 文档i和j有多相似?
  - 术语i和文档j有多相关?

	d_1	d_2	d_3	d_4	d_5	d_6
cosmonaut	1	0	1	0	0	0
astronaut	0	1	0	0	0	0
moon	1	1	0	0	0	0
car	1	0	0	1	1	0
truck	0	0	0	1	0	1

奇异值分解应用

- SVD分解：

$$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^{\mathsf{T}}$$

$A =$

	d_1	d_2	d_3	d_4	d_5	d_6
cosmonaut	1	0	1	0	0	0
astronaut	0	1	0	0	0	0
moon	1	1	0	0	0	0
car	1	0	0	1	1	0
truck	0	0	0	1	0	1

$T =$

cosm.	-0.44	-0.30	0.57	0.58	0.25
astro.	-0.13	-0.33	-0.59	0	0.73
moon	-0.48	-0.51	-0.37	0	-0.61
car	-0.70	0.35	0.15	-0.58	0.16
truck	-0.26	0.65	-0.41	0.58	-0.09

$S =$

2.16	0	0	0	0
0	1.59	0	0	0
0	0	1.28	0	0
0	0	0	1	0
0	0	0	0	0.39

$D^{\mathsf{T}} =$

d_1	d_2	d_3	d_4	d_5	d_6
-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
-0.29	-0.53	-0.19	0.63	0.22	0.41
0.28	-0.75	0.45	-0.20	0.12	-0.33
0	0	0.58	0	-0.58	0.58
-0.53	0.29	0.63	0.19	0.41	-0.22

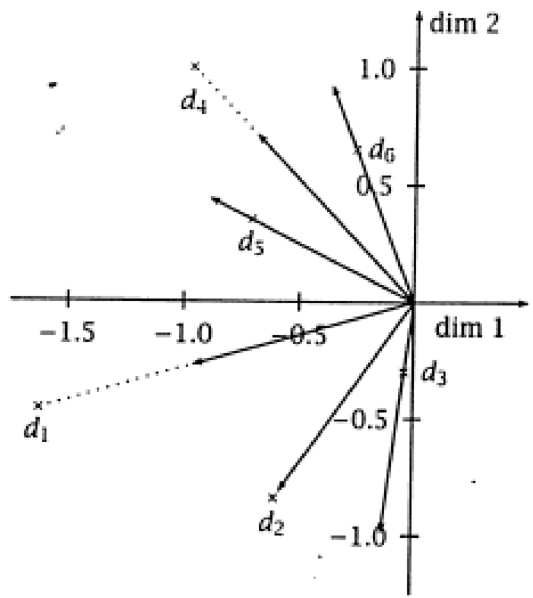
奇异值分解应用

- *A降维处理：* $B=S_{2*2}D^{T}{}_{2*d}$

$X^TU = (U\Lambda V^T)^TU=V\Lambda$

	d_1	d_2	d_3	d_4	d_5	d_6
Dimension 1	−1.62	−0.60	−0.04	−0.97	−0.71	−0.26
Dimension 2	−0.46	−0.84	−0.30	1.00	0.35	0.65

- 图示：



奇异值分解应用

- 向量夹角余弦值：

$$\text{CosSim}(D_i, Q) = \frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^2 \bullet \sum_{k=1}^{t} q_k^2}}$$

- 文本之间相似度矩阵

	d_1	d_2	d_3	d_4	d_5	d_6
d_1	1.00					
d_2	0.78	1.00				
d_3	0.40	0.88	1.00			
d_4	0.47	−0.18	−0.62	1.00		
d_5	0.74	0.16	−0.32	0.94	1.00	
d_6	0.10	−0.54	−0.87	0.93	0.74	1.00

降维前后的对比

- 在新空间中，d_1和d_2之间的相似度为0.78，d_4,d_5和d_6为0.94，0.93，0.74,而在原空间上两者的值是相等的
- 在原空间中，d_2,d_3没有共同的单词，相似度为0，但是在新空间中的相似度为0.88之所已有这种结果，在于它们之间存在着同现模式

	d_1	d_2	d_3	d_4	d_5	d_6
cosmonaut	1	0	1	0	0	0
astronaut	0	1	0	0	0	0
moon	1	1	0	0	0	0
car	1	0	0	1	1	0
truck	0	0	0	1	0	1

	d_1	d_2	d_3	d_4	d_5	d_6
d_1	1.00					
d_2	0.78	1.00				
d_3	0.40	0.88	1.00			
d_4	0.47	−0.18	−0.62	1.00		
d_5	0.74	0.16	−0.32	0.94	1.00	
d_6	0.10	−0.54	−0.87	0.93	0.74	1.00

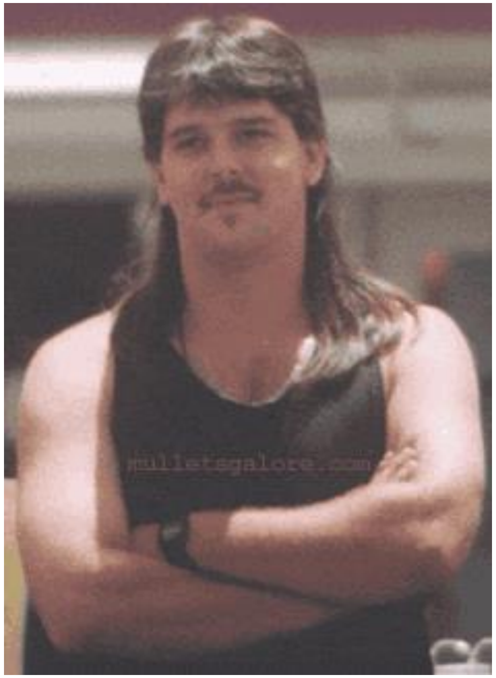
查询处理

- 如何在降维空间中表示查询字段和新增文档
  - 查询可以作为一个伪文档
- 每次重新计算SVD，计算量太大

- 解决方案：

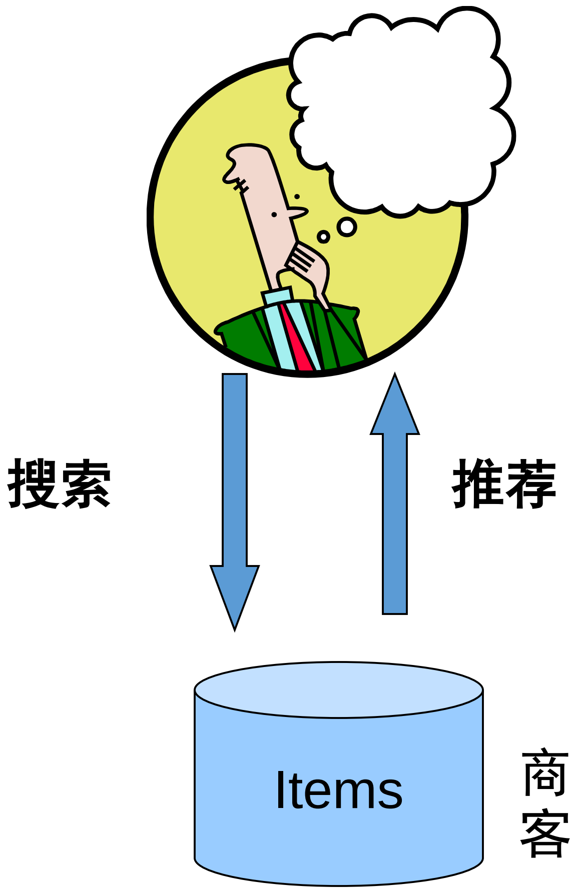
$$A=TSD^T,\ T^TA=T^TTSD^T=SD^T$$

- 新的查询q，再降维后新空间表示为 $T_{t*k}{}^Tq$（可以理解为一种映射）



- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer **X**



商品、网页、博客、网页 …

实例：

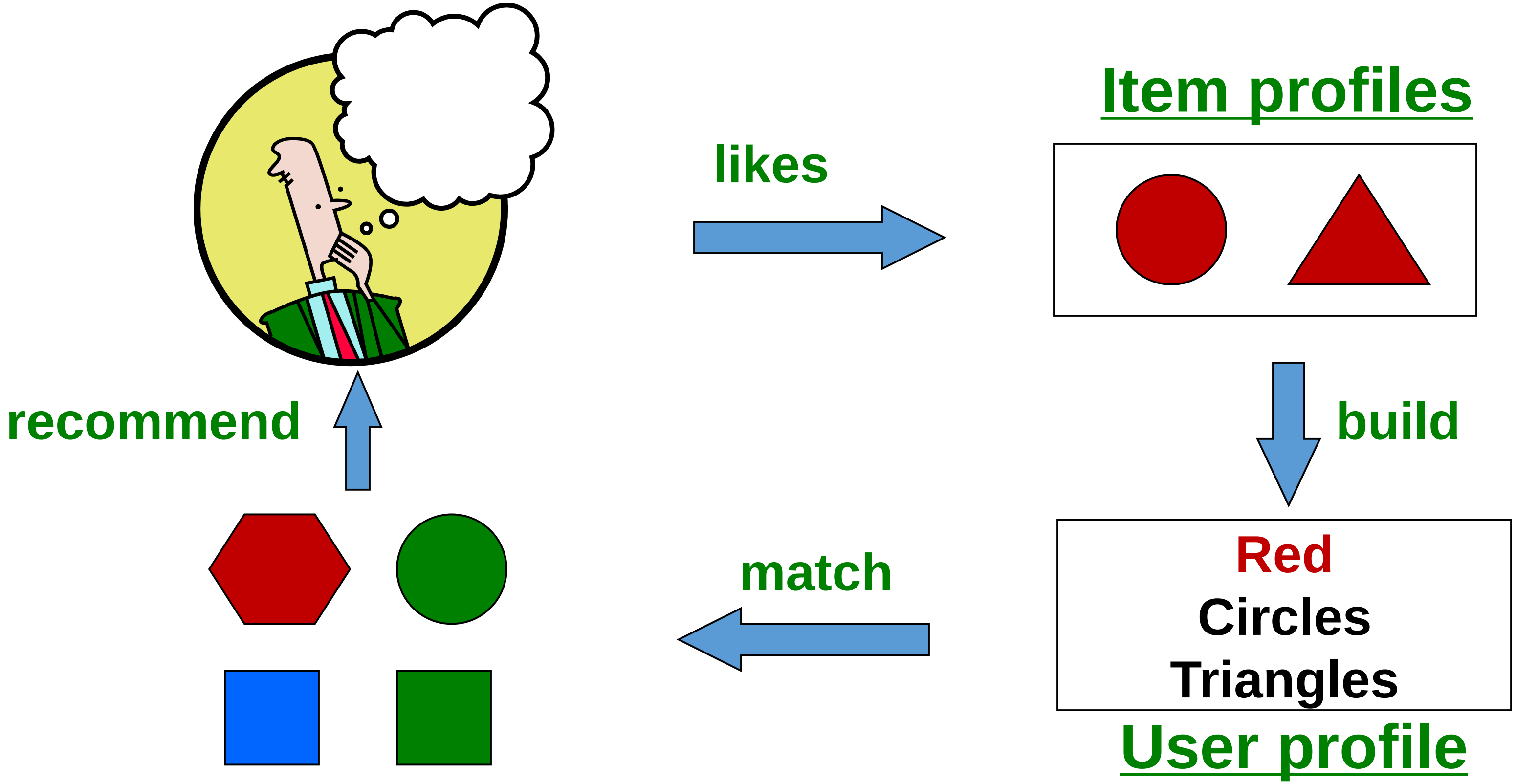
推荐模型

- X = 用户集合，S = 商品集合
- 效用函数u：X × S → R
  - R = 用户评价
  - e.g., 0-5星、[0,1]评分等

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

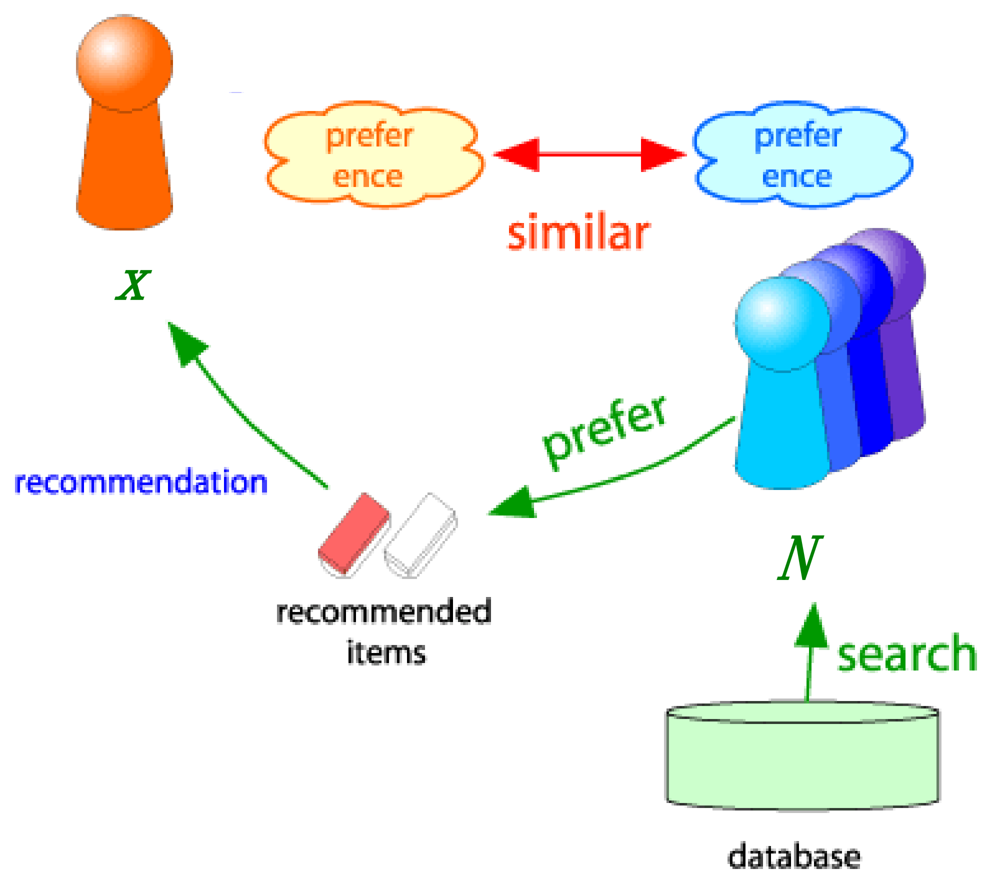
基于内容的推荐

- 基本思想：向用户x推荐和该用户评价较高的相似商品
  - 电影推荐：推荐具有相同演员、导演等内容的电影
  - 网页、博客、新闻推荐：推荐具有相似内容的网站

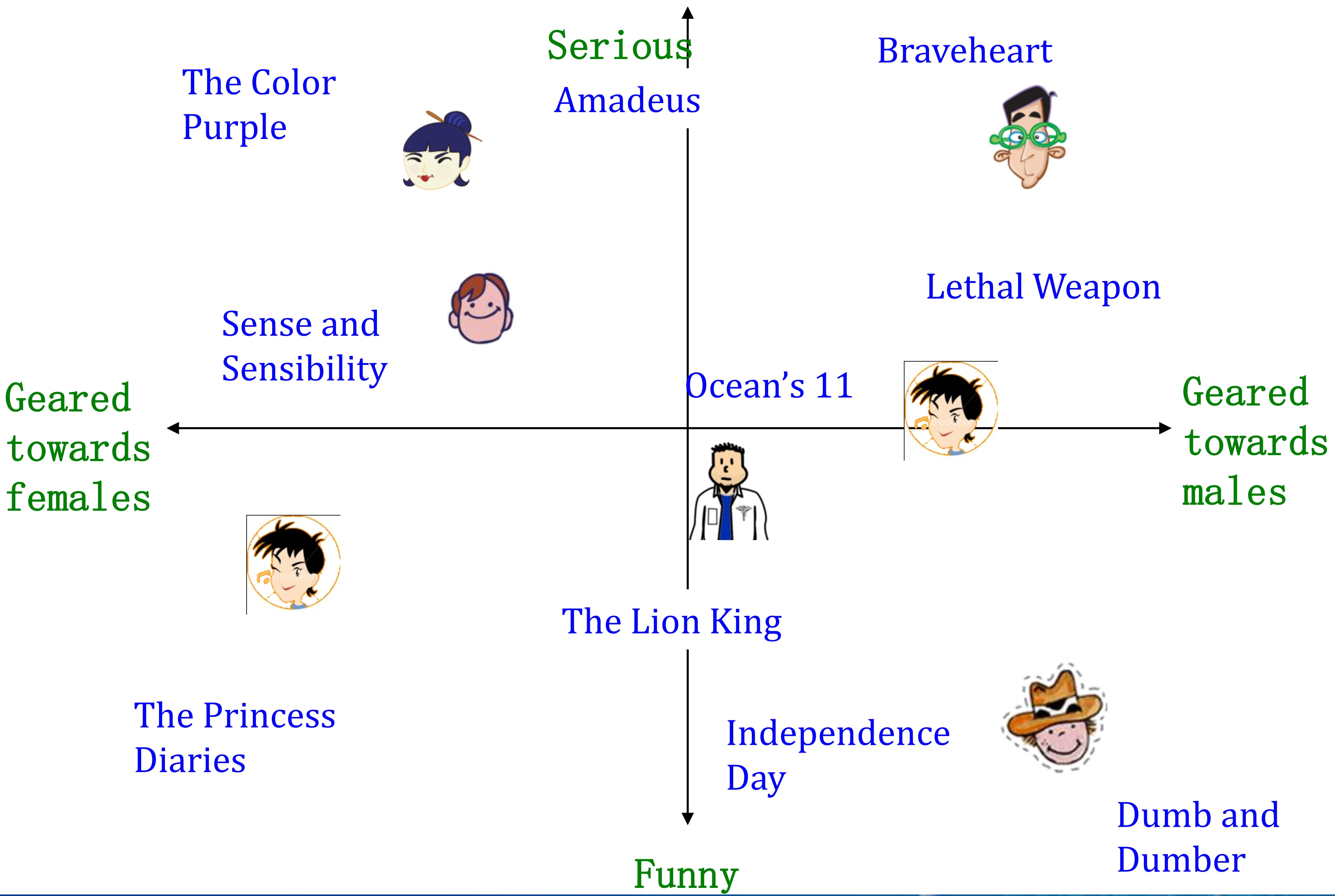


协同过滤（Collaborative Filtering）

- 对于用户x
- 选中其他N个用户，这些用户和x给出的商品评价比较相似
- 基于这些用户的商品评价估计x未评价的商品，并推荐



潜在语义模型（Latent Factor Models)



潜在语义模型

- 采用SVD对评价矩阵进行近似: $R \approx Q \cdot P^T$ **SVD:** $A = U \Sigma V^T$

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

R

$\approx$

factors

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

Q

users

factors

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6

P^T

- SVD基于完整矩阵进行分解，如果评价不完整如何分解?

缺失评价的估计

- 如果存在特定的语义分解，那么如何估计评价

$$\hat{r}_{xi} = q_i \cdot p_x = \sum_f q_{if} \cdot p_{xf}$$

users (columns) × items (rows):

1		3			5			5		4	
		5	4	?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

≈

q_i = row i of Q
p_x = column x of P^T

Q — items (rows) × factors (columns):

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

P^T — factors (rows) × users (columns):

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

潜在语义模型

- 优化目标：寻找P 和 Q

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(\mathbf{i},\mathbf{x})\in \mathbf{R}} (\mathbf{r}_{\mathbf{xi}} - \mathbf{q}_{\mathbf{i}} \cdot \mathbf{p}_{\mathbf{x}})^2$$

  - 为防止过拟合，通常采用

$$\min_{P,Q} \underbrace{\sum_{training} (r_{xi} - q_i p_x)^2}_{\text{“error”}} + \underbrace{\left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 \right]}_{\text{“length”}}$$

- 优化方法：梯度下降
  - 初始化：采用SVD初始化***P***和***Q***
  - 交替优化

$$\boldsymbol{P} \leftarrow \boldsymbol{P} - \eta \bullet \nabla \mathbf{P};\quad \boldsymbol{Q} \leftarrow \boldsymbol{Q} - \eta \bullet \nabla \mathbf{Q}$$

软件开发环境国家重点实验室
State Key Laboratory of Software Development Environment

本节课结束