第四章 降维

4.1 为什么要降维

4.2 主成分分析(PCA)

4.2.1 目标函数

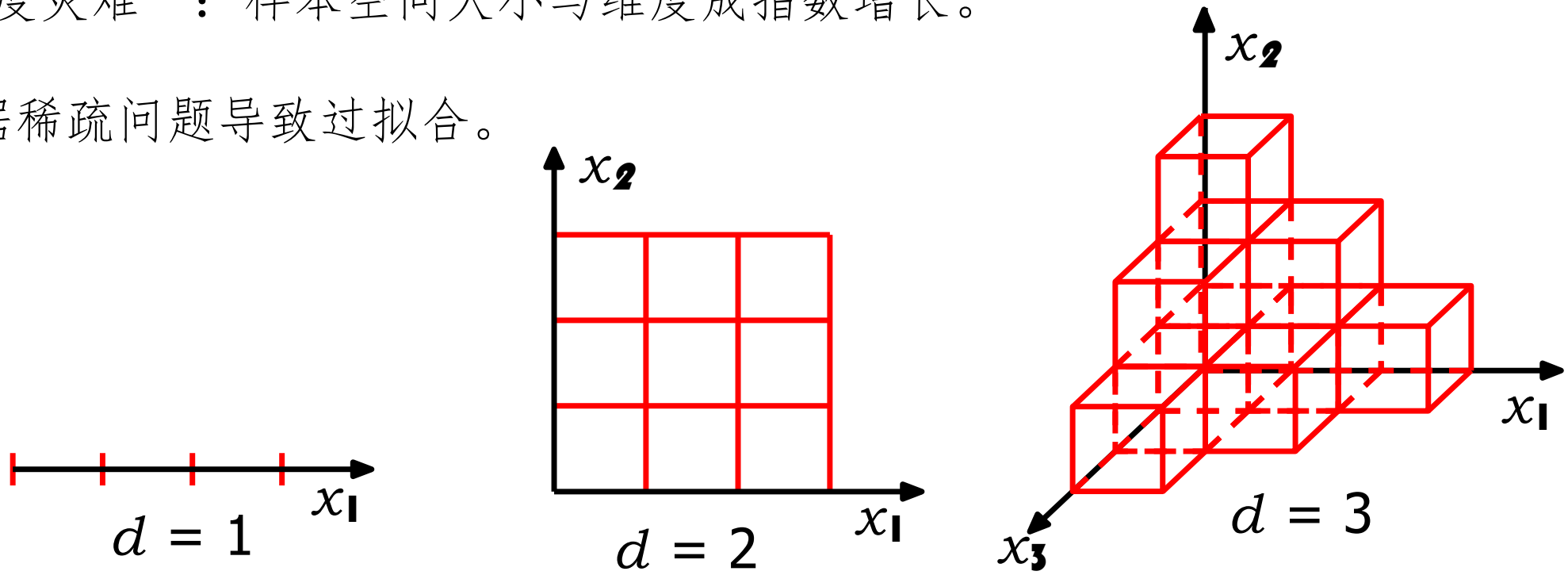
4.2.2 算法

为什么要降维(dimension reduction)?

原始特征太多(高维空间)导致：计算量和占用存贮大、容易过拟合

“维度灾难”：样本空间大小与维度成指数增长。

数据稀疏问题导致过拟合。



为什么要降维(dimension reduction)?

目的:
1.数据压缩，减小存储大小，减小计算量;
2.更有利于后续处理（分类、聚类），防止过拟合;
3.方便可视化。

减少特征数目的方式：

- 特征选择：从给定特征集中选择部分最有用的特征
- 特征提取：基于当前数据集产生新的特征

降维又叫低维“嵌入”（embedding），指从原始的高维空间降到低维空间表示，使得映射后的低维表示尽量保留原数据的基本特性。

选择什么样的映射？

映射后得到的低维表示与原始高维表示之间的误差最小或保留信息最多：

- 尽量保留重要的/主要的维度，丢弃不重要的/次要的维度；
- 尽量保留样本之间的相似性关系：如在原始空间，样本1与样本2的距离很近，与样本3的距离很远。那么在低维空间里也应该保留以上关系。

主要方法：主成分分析(PCA)、多维尺度变化(MDS)等。

第四章 降维

4.1 为什么要降维

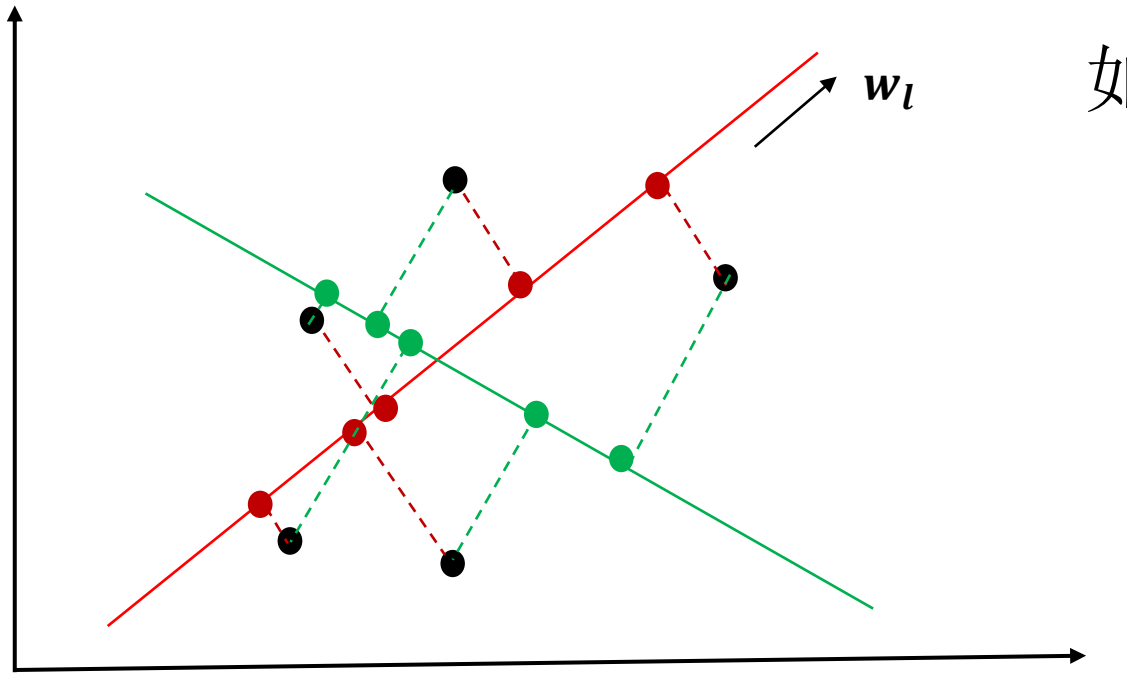
4.2 主成分分析(PCA)

4.2.1 目标函数

4.2.2 算法

主成分分析（Principle Component Analysis）

目标：把数据从原始的高维空间映射到一个低维空间，使得映射后的样本之间具有最大可分性。

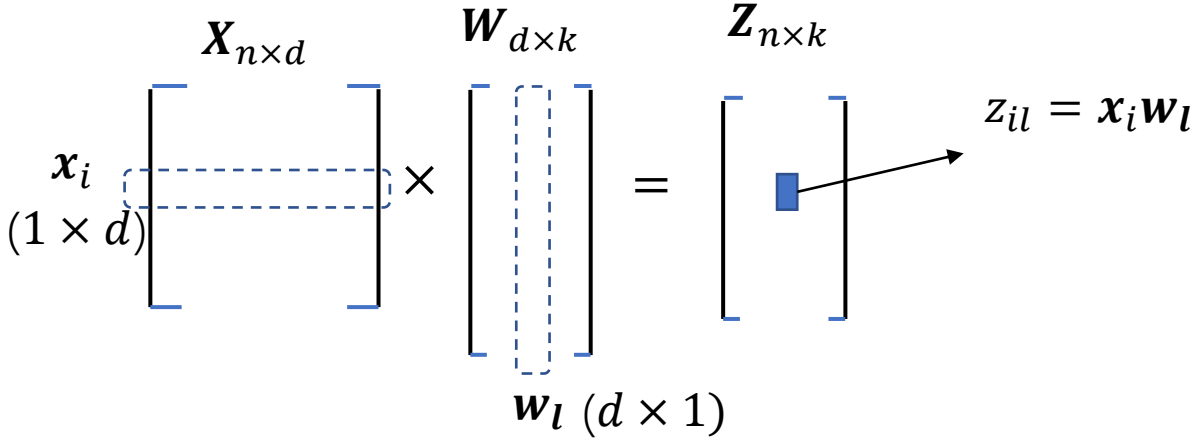


如何衡量可分性？

方差

低维投影下方差的计算

假设数据集表示为对象-属性矩阵$\boldsymbol{X}_{n\times d}$，通过矩阵$\boldsymbol{W}_{d\times k}$把样本$\boldsymbol{x}_i$线性映射到$k$维向量$\boldsymbol{z}_i$。



映射后在第l个维度的方差：

$$\frac{1}{n}\sum_{i=1}^{n}(z_{il}-\bar{z}_l)^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i\boldsymbol{w}_l-\bar{\boldsymbol{x}}\boldsymbol{w}_l)^2 = \boldsymbol{w}_l^T\boldsymbol{C}\boldsymbol{w}_l$$

其中，$\boldsymbol{C}=\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i-\bar{\boldsymbol{x}})^T(\boldsymbol{x}_i-\bar{\boldsymbol{x}})$ 为$\boldsymbol{X}$的协方差矩阵。

方差、协方差的无偏估计用系数$\frac{1}{n-1}$，但是这个系数不影响$\boldsymbol{w}_l$的解。

方差最大化

主成分分析可以建模为最大化方差，同时满足映射向量为单位向量的条件，即：

$$\max \boldsymbol{w_l^T C w_l}$$

$$\text{s.t. } \boldsymbol{w_l^T w_l} = 1$$

以上问题是一个带约束条件的求极值问题，用拉格朗日乘子法

$$\mathrm{L}(\boldsymbol{w_l}, \lambda) = \boldsymbol{w_l^T C w_l} + \lambda(1 - \boldsymbol{w_l^T w_l})$$

令 $\frac{\boldsymbol{\partial}\mathrm{L}(\boldsymbol{w_l},\lambda)}{\boldsymbol{\partial w_l}} = 2\boldsymbol{C w_l} - 2\lambda \boldsymbol{w_l} = 0$

由以上公式得到

$$\boldsymbol{C w_l} = \lambda \boldsymbol{w_l}$$

即$\boldsymbol{w_l}$是矩阵 $\boldsymbol{C}$的特征向量，对应特征值为λ。

规范化后进行PCA

假设数据$\boldsymbol{X_{n\times d}}$**已经零均值规范化(每列均值**$\bar{\boldsymbol{x}}$**=0)，** 那么$\bar{\boldsymbol{x}}\,\boldsymbol{w_l}$ =0

映射后在第l个维度的方差退化为:

$$\frac{1}{n}\sum_{i=1}^{n}(z_{il}-\bar{z_l})^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i w_l}-\bar{\boldsymbol{x}}\boldsymbol{w_l})^2$$

$$\frac{1}{n}\sum_{i=1}^{n} z_{il}{}^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i w_l})^2 = \boldsymbol{w_l^T C w_l}$$

为什么PCA前一般要规范化?

其中， $\boldsymbol{C=\frac{1}{n}X^TX}$ 为规范化后的协方差矩阵。

推导： $\bar{z}_l = \bar{\boldsymbol{x}}\boldsymbol{w}_l$

$\bar{\boldsymbol{x}}$是$\boldsymbol{X}$按列求均值得到的横向量;

$\bar{z}_l = \frac{1}{n}\sum_{i=1}^{n} z_{il}$为$\boldsymbol{Z}$第$l$列的均值。

PCA算法

输入：包含n个d维样本的数据集$\boldsymbol{X}_{n\times d}$、输出维度大小k.

输出：$\boldsymbol{W}=[w_1,w_2,\dots w_k]$, 其中$w_l$ 为第l个特征向量，对应第l大的特征值。

步骤：

1.计算协方差矩阵$\boldsymbol{C}_{p\times p}$;

2. 计算$\boldsymbol{C}$的k个最大特征值的特征向量;

3. 由得到的特征向量输出映射矩阵$\boldsymbol{W}$;

什么时候计算量大？
有没有其他方法得到$\boldsymbol{C}$的特征向量？

求解协方差矩阵$\boldsymbol{C}$的特征向量

不用计算$\boldsymbol{C}$而直接对$\boldsymbol{X}$进行奇异值分解(Singular Value Decomposition)。

SVD把矩阵$\boldsymbol{X}_{n\times d}$分解成两个正交矩阵与一个对角矩阵$\boldsymbol{\Sigma}$：$\boldsymbol{X} = \boldsymbol{U\Sigma V^T}$,
其中

$\boldsymbol{\Sigma}_{r\times r} = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix}$，$\sigma_i$为$\boldsymbol{X}$的奇异值，

左奇异向量$\boldsymbol{U}_{n\times r} = [\boldsymbol{u}_1, \ldots \boldsymbol{u}_r]$为$(\boldsymbol{XX^T})_{\boldsymbol{n\times n}}$的特征向量，

右奇异向量$\boldsymbol{V}_{d\times r} = [\boldsymbol{v}_1, \ldots \boldsymbol{v}_r]$为$(\boldsymbol{X^T X})_{d\times d}$的特征向量，**即$\boldsymbol{C}$的特征向量。**

假设$\sigma_1 > \sigma_2 \cdots > \sigma_r$,则$\sigma_k = \sqrt{\lambda_k}$,$\sigma_k$对应的向量$\boldsymbol{v}_k$即为对应$\boldsymbol{C}$的$k$大特征值$\lambda_k$的特征向量$\boldsymbol{w}_k$,即$\boldsymbol{v_k} = \boldsymbol{w_k}$，。

降维后的表示与重构

对$\boldsymbol{X}_{n\times d}$进行PCA降维后得到正交矩阵$\boldsymbol{W}_{d\times k}$，则样本在这个低维空间的表示为$\boldsymbol{Z}_{n\times k}=\boldsymbol{XW}$。基于这个表示对原始数据进行重构，即映射回高维空间，得到$\boldsymbol{X'}_{n\times d}=\boldsymbol{ZW}^T=\boldsymbol{XWW}^T$。重构误差为

$$e=\|\boldsymbol{X}-\boldsymbol{X'}\|_2^2=\sum_{i=1}^{n}\sum_{i=1}^{n}(x_{ij}-x'_{ij})^2$$

若对$\boldsymbol{X}$进行SVD分解后取前面最大的k个奇异值对应的$\boldsymbol{U}$，$\boldsymbol{\Sigma}$，$\boldsymbol{V}$，则$\boldsymbol{X'}_{n\times d}=\boldsymbol{U}_{n\times k}\boldsymbol{\Sigma}_{k\times k}\boldsymbol{V}^T_{k\times d}$，对应有$\boldsymbol{W}=\boldsymbol{V}$，$\boldsymbol{Z}_{n\times k}=\boldsymbol{U\Sigma}$。

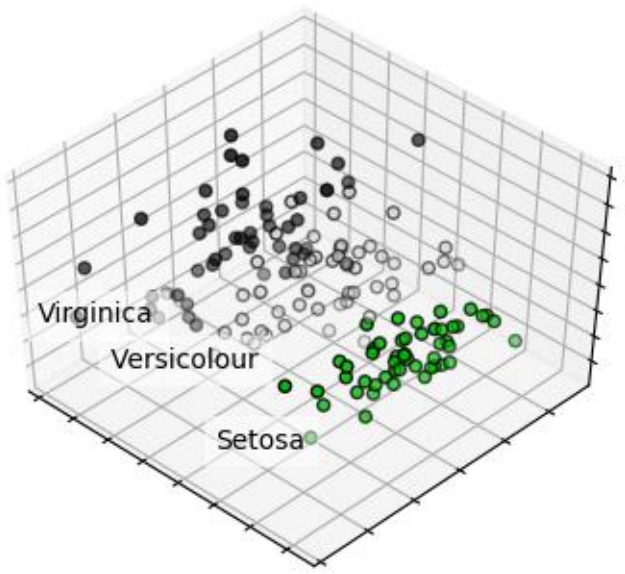
Python实现SVD

调用scipy.linalg.svd

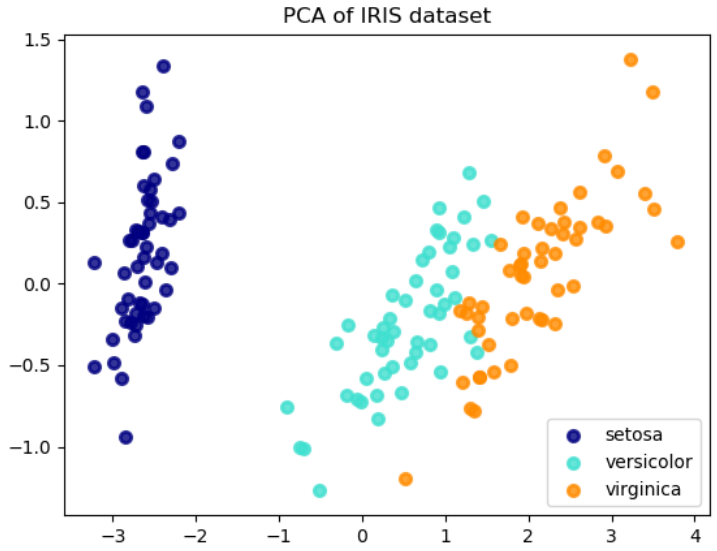
```
from scipy import linalg
n,d=9,6
x=np.random.rand(n,d)
#n,d=x.shape
#S为r个奇异值组成的向量（r=min(n,d)）
U, s, Vh = linalg.svd(x, full_matrices=False)
#转成对角矩阵
sigma = np.diag(s)
#基于分解结果重构矩阵，计算重构误差
x1 = np.dot(U, np.dot(sigma, Vh))
err=((x-x1)*(x-x1)).sum()
k=4
U_k=U[:,:k]
sigma_k=sigma[:k,:k]
Vh_k=Vh[:k]
x2 = np.dot(U_k, np.dot(sigma_k, Vh_k))
err_k =((x-x2)*(x-x2)).sum()
```

PCA降维后可视化

- 由PCA算法得到映射矩阵$\boldsymbol{W}$后，计算映射后的数据$\boldsymbol{Z}_{n\times k}=\boldsymbol{XW}$。



三个主成分



两个主成分