CSDN
不 止 于 代 码

4.3 PCA

CSDN学院

2017年10月

大纲

- PCA原理
- Scikit learn中的PCA实现
- PCA案例分析

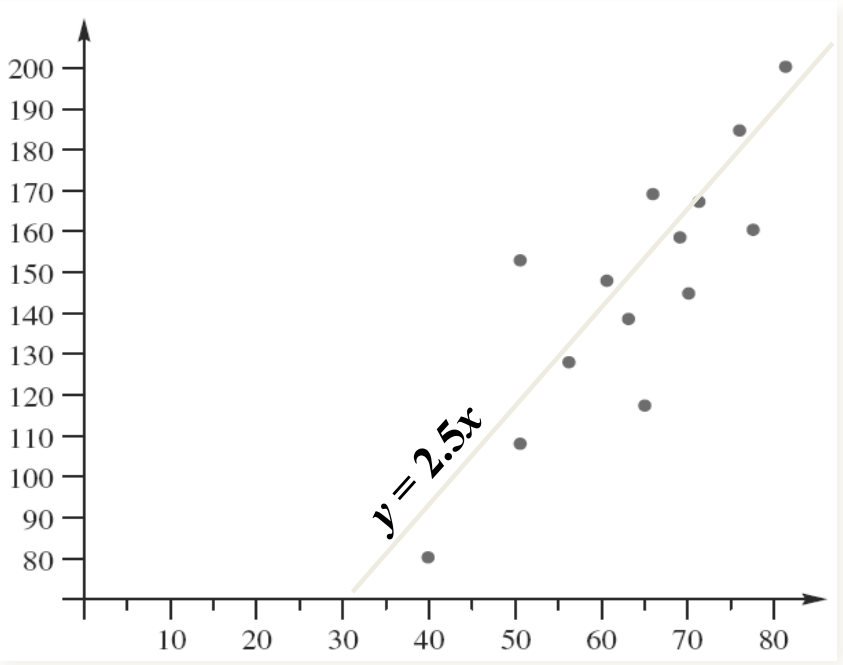
主成分分析

- 主成分分析（Principal Components Analysis, PCA）是由Hotelling于1933年首先提出，亦被称为Karhunen-Loéve变换(KLT)。
- 动机：多个变量之间往往存在着一定程度的相关性，可以通过线性组合的方式，从其中提取信息。
- 主成分分析：将原始的D维数据投影到低维空间，并尽可能保留更多信息
  - 投影后的方差最大
  - 最小化重构平方误差

  } 二者等价
- 从而达到降维的目的：用较少的主成分得到较多信息

► 例：原始数据

Height	Weight
65	170
75	188
60	150
70	170
56	130
80	203
68	160
50	110
40	80
50	153
69	148
62	140
76	164
64	120



坐标旋转

- 考虑（可逆）变换

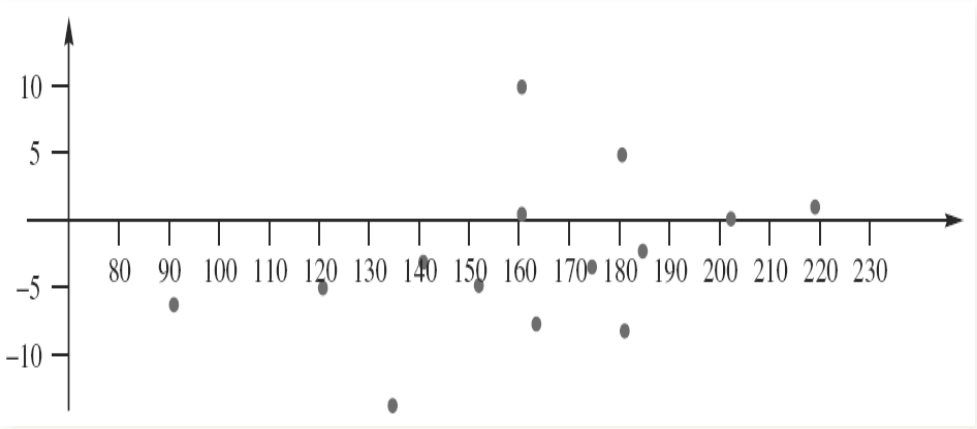
$$\boldsymbol{\theta} = \mathbf{A}\mathbf{x}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}, \phi = \arctan 2.5$$

$$\mathbf{A} = \begin{bmatrix} 0.37139068 & 0.92847669 \\ -0.92847669 & 0.37139068 \end{bmatrix}$$

变换后的序列

First Coordinate	Second Coordinate
182	3
202	0
162	0
184	−2
141	−4
218	1
174	−4
121	−6
90	−7
161	10
163	−9
153	−6
181	−9
135	−15



降维

- 抛弃坐标第二维…，维度可降低50%！

例：重构序列

Height	Weight
65	170
75	188
60	150
70	170
56	130
80	203
68	160
50	110
40	80
50	153
69	148
62	140
76	164
64	120

原始值

Height	Weight
68	169
75	188
60	150
68	171
53	131
81	203
65	162
45	112
34	84
60	150
61	151
57	142
67	168
50	125

重构值

例：误差分析

$$\{\hat{x}_i\} \equiv \text{重构序列}$$

$$\hat{\theta}_i = \begin{cases} \theta_i & i = 0, 2, 4, \ldots \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_{i=1}^{N} (x_i - \hat{x}_i)^2 = \sum_{i=1}^{N} (\theta_i - \hat{\theta}_i)^2$$

- 误差取决于被置为0的那些θ_i的幅值
  - 如果幅值很小，则误差也很小
  - 即大多数信息在每个数据对的第一个元素中

PCA算法

- 给定数据 $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$，计算协方差矩阵 $\boldsymbol{\Sigma}$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

- PCA 的基向量 = $\boldsymbol{\Sigma}$ 的特征向量

- 大的特征值 ⇒ 更重要的特征向量

PCA 算法

PCA algorithm($\mathbf{X}$, K): 前K 个特征值/特征向量

$\mathbf{X}$ 为$D \times N$ 数据矩阵（注意和一般我们记X为 $N \times D$的矩阵），N为样本数目，D为特征维数
每个数据点 $\mathbf{x}_i$ = 列向量, i=1..N

- 计算均值 $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ （每维特征的均值）
- $\mathbf{X} \leftarrow$ 每个列向量$\mathbf{x}_i$ 减去均值 $\bar{\mathbf{x}}$（数据中心化）
- 计算协方差矩阵 $\Sigma \leftarrow \mathbf{X}\,\mathbf{X}^{\mathrm{T}}$
- 计算Σ的特征值/特征向量$\{\lambda_j, \mathbf{u}_j\}_{j=1..D}$，且... $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$
- Return $\{\lambda_j, \mathbf{u}_j\}_{j=1..K}$ #前K个主成分

►证明：最小损失

- 令$\mathbf{x} \in \mathbb{R}^D$为$D$维向量
- $\mathbf{U} = \begin{pmatrix} \mathbf{u}_1^T \\ \dots \\ \mathbf{u}_D^T \end{pmatrix} \in \mathbb{R}^{D \times D}$ 为正交矩阵，$\mathbf{U}\mathbf{U}^T = \mathbb{I}_D$
- $\mathbf{y} \doteq \mathbf{U}\mathbf{x}, \quad \mathbf{x} = \mathbf{U}^T\mathbf{y} = \sum_{j=1}^{D} \mathbf{u}_j y_j$
- 则重构为 $\hat{\mathbf{x}} \doteq \sum_{j=1}^{K} \mathbf{u}_j y_j \quad (K < D)$，即用$K$个基向量近似$\mathbf{x}$
- 重构误差为 $\varepsilon^2 = \mathbb{E}\left(\|\mathbf{x} - \hat{\mathbf{x}}\|^2\right) = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$
- 所以PCA的目标函数为：$\arg\min_{\mathbf{U}} \varepsilon^2, s.t.\ \mathbf{U}\mathbf{U}^T = \mathbb{I}_D$

证明：最小损失

$$\mathbf{y} \doteq \mathbf{U}\mathbf{x}, \quad \mathbf{x} = \mathbf{U}^T\mathbf{y} = \sum_{j=1}^{D} \mathbf{u}_j y_j$$

- $$\varepsilon^2 = \mathbb{E}\left(\|\mathbf{x} - \hat{\mathbf{x}}\|^2\right) = \mathbb{E}\left(\left\|\sum_{j=1}^{D} \mathbf{u}_j y_j - \sum_{j=1}^{K} \mathbf{u}_j y_j\right\|^2\right)$$

$$= \mathbb{E}\left(\sum_{j=K+1}^{D} y_j \mathbf{u}_j^T \mathbf{u}_j y_j\right) = \mathbb{E}\left(\sum_{j=K+1}^{D} y_j^2\right)$$

$$= \sum_{j=K+1}^{D} \mathbb{E}\left(\left(\mathbf{u}_j^T \mathbf{x}\right)\left(\mathbf{x}^T \mathbf{u}_j\right)\right)$$

$$= \sum_{j=K+1}^{D} \mathbf{u}_j^T \mathbb{E}\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{u}_j$$

$$= \sum_{j=K+1}^{D} \mathbf{u}_j^T \boldsymbol{\Sigma} \mathbf{u}_j$$ （因为x已经中心化）

►证明：最小损失

- PCA目标为：$\arg\min_{\mathbf{u}_{K+1},\ldots,\mathbf{u}_D} \varepsilon^2,\ s.t.\ \mathbf{U}\mathbf{U}^T = \mathbb{I}_D$
- 用拉格朗日乘子法，得到

$$L(\mathbf{U},\boldsymbol{\lambda}) = \varepsilon^2 - \sum_{j=K+1}^{D} \lambda_j \left(\mathbf{u}_j^T \mathbf{u}_j - 1\right)$$

$$= \sum_{j=K+1}^{D} \mathbf{u}_j^T \boldsymbol{\Sigma} \mathbf{u}_j - \sum_{j=K+1}^{D} \lambda_j \left(\mathbf{u}_j^T \mathbf{u}_j - 1\right)$$

$$\frac{\partial L(\mathbf{U},\boldsymbol{\lambda})}{\partial \mathbf{u}_j} = 2\boldsymbol{\Sigma}\mathbf{u}_j - 2\lambda_j \mathbf{u}_j = 0$$

▶证明：最小损失

$$\frac{\partial L(\mathbf{U},\boldsymbol{\lambda})}{\partial \mathbf{u}_j} = 2\boldsymbol{\Sigma}\mathbf{u}_j - 2\lambda_j\mathbf{u}_j = 0 \qquad \Rightarrow \boldsymbol{\Sigma}\mathbf{u}_j = \lambda_j\mathbf{u}_j$$

- 即$\left[\mathbf{u}_j, \lambda_j\right]$为$\boldsymbol{\Sigma}$的特征向量和特征值

$$\varepsilon^2 = \sum_{j=K+1}^{D} \mathbf{u}_j^T\boldsymbol{\Sigma}\mathbf{u}_j = \sum_{j=K+1}^{D} \mathbf{u}_j^T\lambda_j\mathbf{u}_j = \sum_{j=K+1}^{D} \lambda_j$$

- 因此当 $\lambda_{K+1},\ldots,\lambda_D$ 为$\boldsymbol{\Sigma}$的最小的特征值且 $\mathbf{u}_{K+1},\ldots,\mathbf{u}_D$为对应的特征向量时，重构残差$\varepsilon^2$最小。

▶证明：最大投影后方差

- 假设u为投影方向，则x投影后为 $\mathbf{y} = \mathbf{u}^T \mathbf{x},$
- 投影后的方差为

$$\sum_{i=1}^{N} (\mathbf{y}_i - \bar{y})^2 = \sum_{i=1}^{N} (\mathbf{u}^T \mathbf{x}_i - \mathbf{u}^T \bar{\mathbf{x}})^2 = \mathbf{u}^T \boldsymbol{\Sigma} \mathbf{u}$$

- 即目标函数为：$\arg \max_{\mathbf{u}} \mathbf{u}^T \boldsymbol{\Sigma} \mathbf{u}, \quad s.t.\ \mathbf{u}^T \mathbf{u} = 1$
- 采用拉格朗日乘子法，得到拉格朗日函数为

$$L(\mathbf{u}, \lambda) = \mathbf{u}^T \boldsymbol{\Sigma} \mathbf{u} - \lambda (\mathbf{u}^T \mathbf{u} - 1)$$

- 求偏导，得到

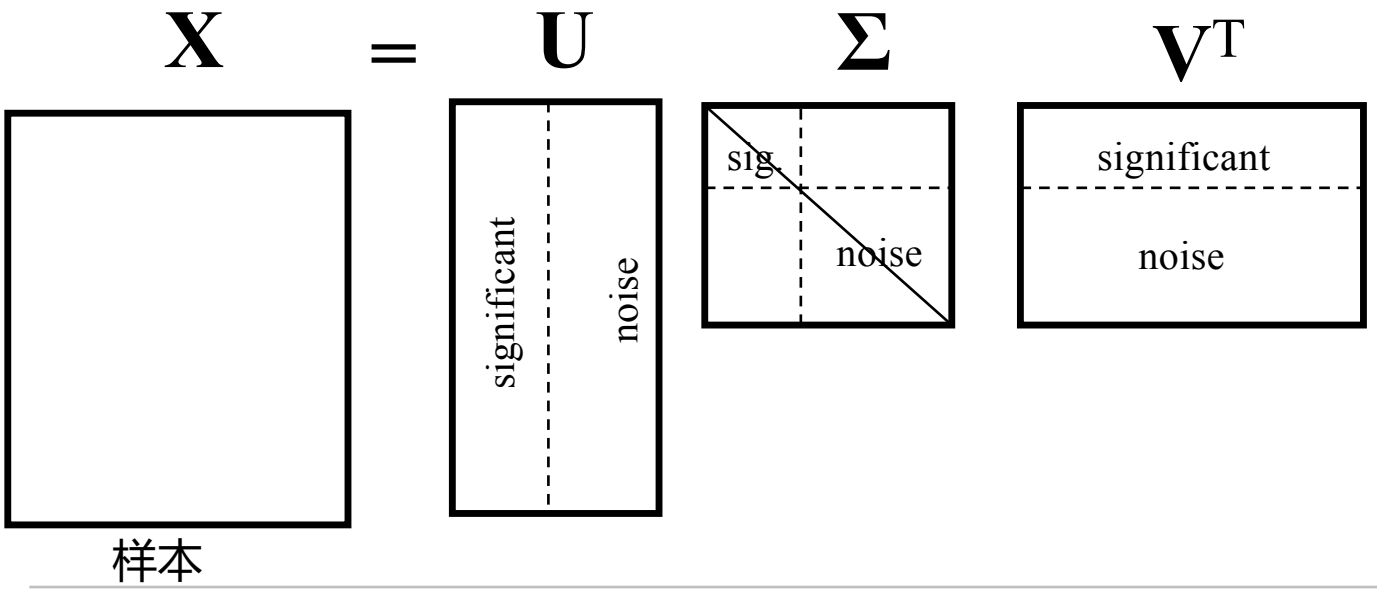
$$\frac{\partial L(\mathbf{u}, \lambda)}{\partial \mathbf{u}} = 2\boldsymbol{\Sigma}\mathbf{u} - 2\lambda\mathbf{u} = 0 \quad \Rightarrow \boldsymbol{\Sigma}\mathbf{u} = \lambda\mathbf{u}$$

当λ 为Σ的最大的特征值且u为对应的特征向量时，投影后的方差最大。

PCA算法: SVD

- 也可以对中心化后的数据矩阵**X**进行SVD分解实现PCA

$$\mathbf{X}_{N\times D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$$



回忆：SVD分解

左奇异向量　奇异值矩阵　右奇异向量

- X的SVD分解：$\mathbf{X}_{N\times D} = \mathbf{U}_{N\times N}\mathbf{S}_{N\times D}\mathbf{V}_{D\times D}^{T}$
- 将矩阵X的转置$\mathbf{X}^{\mathrm{T}} * \mathbf{X}$，将会得到一个方阵，对这个方阵求特征值（特征值只对方阵有意义）：

$$\left(\mathbf{X}_{D\times N}^{T}\mathbf{X}_{N\times D}\right)\mathbf{v}_j = \lambda_j \mathbf{v}_j$$

- 这里得到的v，就是上面的右奇异向量。
- 此外，还可以得到：$\sigma_j = \sqrt{\lambda_j}$

$$\mathbf{u}_j = \frac{1}{\sigma_j}\mathbf{X}\mathbf{v}_j$$

选择主成分的数目

- 第k个主成分对方差的贡献率为：$\lambda_k \Big/ \sum_{j=1}^{D} \lambda_j$
  - 前k个主成分贡献率的累计值称为累计贡献率。
- 主成分数目通常有两种方式：
  - 直接确定主成分数目
  - 根据主成分的累计贡献率确定主成分数目，如累计贡献率大于85%

大纲

- PCA原理
- Scikit learn中的PCA实现
- PCA-案例

Scikit learn 中PCA的实现

- from sklearn.decomposition import PCA
  - PCA是一种线性降维技术，采用 SVD对数据进行处理，保留最重要的前 *K*个奇异值向量，用较低维度空间对原数据集进行映射
  - PCA 类的实现采用 scipy.linalg 来实现 SVD ，只作用于密集矩阵，并且不能扩展到高维数据。对于 n 维 的 n 个数据，时间复杂度是 O(n^3)。
- PCA的构造函数：

```
sklearn.decomposition.PCA(n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None)
```

PCA参数

- n_components：int 或者 string
  - 缺省: None，所有成分被保留
  - int: 要保留的主成分个数K
  - String:如n_components='mle'，将自动选取特征个数K，使得满足所要求的方差百分比
- copy：True或者False，表示在运行算法时是否将原始训练数据复制一份
  - 缺省时默认为True
  - True: 将保持原始数据不变，False 则直接在原始数据上进行计算
- whiten：白化，是否使得每个特征具有相同的方差
  - 缺省: False

PCA参数

- svd_solver： PCA实现算法
  - auto : 根据数据X的形状和主成分的数目自动选择实现算法
    - 如果输入数据大于500x500 且主成分数目小于80% 的维数（N_{min}=min($n_{samples}$, $n_{features}$)），采用更有效的 'randomized' 方法
    - 其他情况：先计算全部SVD，然后截断
  - full：调用标准的LAPACK计算全部SVD（ scipy.linalg.svd ），然后截断
  - arpack : 调用ARPACK 计算截断K个成分的SVD（ scipy.sparse.linalg.svds
  - randomized : 采用Halko提出的方法计算randomized SVD
    - 记N_{max}=max($n_{samples}$, $n_{features}$)， N_{min}=min($n_{samples}$, $n_{features}$), randomized SVD的计算复杂度为 $O\left(N_{max}^2 \times n_{components}\right)$
    - 完全PCA的计算复杂度为 $O\left(N_{max}^2 \times N_{min}\right)$

PCA 类包含的属性

- components_ ：array, [n_components, n_features]：主成分
- explained_variance_ratio_ ：array, [n_components]：保留的K个成分各自的方差百分比
- n_components_ ：主成分的个数K
  - 当它被设为’mle’或者0-1的数字时，表示选中方差百分比和的比重
- noise_variance_ ：

使用 PCA 的方法流程

- 通过参数实例化 PCA 类，通常要定义要保留的维数 n_components
- 调用fit()函数对数据进行训练
- 然后调用 transform()方法返回降维后的数据

大纲

- PCA原理
- Scikit learn中的PCA实现
- PCA案例

案例分析

- 手写数字识别

CSDN
不 止 于 代 码

THANK YOU

AI100