机器学习讲义(L16)：主成分分析

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

1 主成分分析 (Principle Component Analysis)

在今天的各种实际应用中，包括互联网领域和生物、医学领域等，机器学习所处理数据的一大特征就是维数非常大，维数超过样本数的应用场景也非常多。大多数机器学习方法都非常难以应对维数过高的情况，此时就需要从原数据中提取最关键的信息，将特征维数降低，从而提高机器学习的性能。

特征选取有很多方法，在之前的课程中也都介绍过了。特征选取的问题是，单纯选取一个特征子集可能会丢失其他特征的有用信息，而若我们将部分特征组合为一个新特征，有可能就能利用好所有的特征信息。选取一系列原特征的加权和作为新特征，这就是降维。降维的一个常用方法是主成分分析 (PCA)。

1.1 目的

降维实际上是一种投影，或称坐标变换，即旋转。在数学上，我们可以方便地将投影定义为内积并给出坐标变换公式：

定义：坐标变换

对任意 $x \in \mathbb{R}^p$，若 $v_i \in \mathbb{R}^p$，$i = 1, 2, \cdots, k (k < p)$ 为 $\mathbb{R}^p$ 的某个子空间的一组标准正交基，则 x 对该基的坐标 u 满足

$$u = (v_1, v_2, \cdots, v_k)^T x = Vx \tag{1}$$

通常地，我们希望数据在各维度上都足够离散，从而更容易将不同的数据点区分开。因此一个直接的想法是希望降维后数据点在各坐标轴上的方差最大。对于降为一维的简单情况，可以认为是寻找数据方差最大的方向。给定已中心化数据集 $X = (x_1, x_2, \cdots, x_n)^T$，若 v 为降维投影方向（一个单位向量），因为数据已经中心化，降维后的样本方差可写为

$$\mathrm{Var}(u) = \frac{1}{n}\sum_{i=1}^{n}(u_i - \bar{u})^2 = \frac{1}{n}\sum_{i=1}^{n} u_i^2 \tag{2}$$

其中 $u_i = v^T x_i$。问题可写为数学形式：

$$\mathrm{argmax} \sum_{i=1}^{n} u_i^2 \tag{3}$$

1.2 求解

我们先将问题 (3) 写为矩阵形式，由于 $(u_1, u_2, \cdots, u_n)^T = (x_1, x_2, \cdots, x_n)^T v = Xv$：

$$\begin{aligned} &\mathrm{argmax}\ v^T X^T X v \\ &\text{s.t. } v^T v = 1 \end{aligned} \tag{4}$$

记 $v^T X^T X v = \lambda$，有

$$\begin{aligned} v^T X^T X v &= \lambda \\ &= \lambda v^T v \\ \Rightarrow v^T (X^T X v - \lambda v) &= 0 \end{aligned} \tag{5}$$

因为 $X^T X$ 是实对称矩阵，一定能找到一组特征向量为标准正交基，假设这组特征向量为 $u_1, u_2, \cdots, u_p$，p 为维数，则

$$X^T X = \sum_{i=1}^{p} \lambda_i u_i u_i^T \tag{6}$$

其中 λ_i 为 u_i 对应的特征值。并且存在唯一的一组 $a_1, a_2, \cdots, a_p$ 使得

$$v = \sum_{i=1}^{p} a_i u_i \tag{7}$$

并且由于 v 为单位向量，$\sum_{i=1}^{p} a_i^2 = 1$。将 (7) 代入 (5)，得到

$$(\sum_{i=1}^{p} a_i u_i)^T (\sum_{i=1}^{p} a_i X^T X u_i - \lambda \sum_{i=1}^{p} a_i u_i) = 0 \tag{8}$$

因为 $X^T X u_i = \lambda_i u_i$，

$$\begin{aligned} (\sum_{i=1}^{p} a_i u_i)^T (\sum_{i=1}^{p} a_i \lambda_i u_i - \lambda \sum_{i=1}^{p} a_i u_i) &= 0 \\ \Rightarrow (\sum_{i=1}^{p} a_i u_i)^T (\sum_{i=1}^{p} a_i (\lambda_i - \lambda) u_i) &= 0 \\ \Rightarrow \sum_{i=1}^{p}\sum_{j=1}^{p} (\lambda_j - \lambda) a_i a_j u_i^T u_j &= 0 \end{aligned} \tag{9}$$

注意到 $u_1, u_2, \cdots, u_p$ 为标准正交基，则有 $u_i^T u_j = 0,\ i \neq j$，$u_i^T u_i = 1$，则

$$\sum_{i=1}^{p} (\lambda_i - \lambda) a_i^2 = 0 \tag{10}$$

移项得

$$\begin{aligned} \lambda &= \frac{\sum_{i=1}^{p} a_i^2 \lambda_i}{\sum_{i=1}^{p} a_i^2} \\ &= \sum_{i=1}^{p} a_i^2 \lambda_i \\ &\le \max(\lambda_i) \end{aligned} \tag{11}$$

最后一个不等号使用了放缩 $\lambda_i \le \max(\lambda_i)$。这表明，问题 (3) 取到最优值即 $X^T X$ 的最大特征值当且仅当 v 为对应的单位特征向量。

下图展示了取最大方差方向投影后的结果。

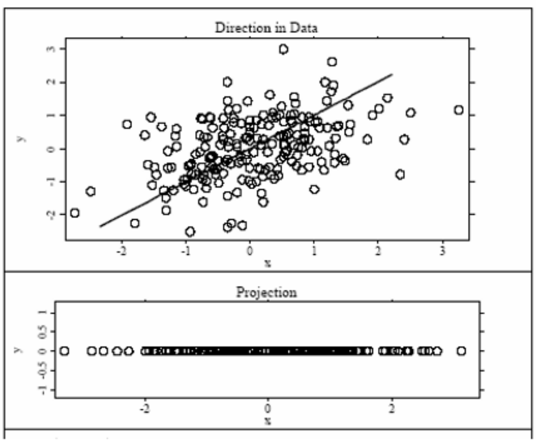

Figure 1: 主成分分析投影示意

2 PCA 在数据降维中的应用

PCA 在实际中有非常多的应用，在数据可视化，数据分类，数据分析等领域拥有一席之地。它最直接的应用就是用来数据降维，同时保留尽量多的信息，可以达到去除噪声和不重要特征的效果。正如前文说到的，PCA 可以找到使样本方差最大的投影方向，然后将每个样本投影到低维的空间里。例如原样本空间有 p 个坐标轴，表示为 $x_1, x_2, \ldots, x_p$，通过 PCA 产生 $k(k < p)$ 个新坐标轴，即协方差矩阵的 k 个特征向量，特征向量之间两两垂直，表示为 $v_1, v_2, \ldots, v_k$，那么可以把样本坐标变换到新坐标系中，实现降维。$v_1, \ldots, v_k$ 也被称之为主成分，将其按照对应的特征值从大到小排列，那么 v_1 方向是使样本投影下来方差最大的方向，v_2 就是使方差第二大的方向，以此类推，v_k 就是使方差第 k 大的方向。这一点很容易证明。

证明. 样本的坐标投影到 v_k 方向的值为 u_k，v_k 对应的特征值为 λ_k 且 v_k 为单位向量，样本矩阵为 $X \in \mathbb{R}^{n \times p}$，那么方差

$$Var(u_k) = v_k^T X^T X v_k = v_k^T \lambda_k v_k = \lambda_k v_k^T v_k = \lambda_k \tag{12}$$

可以看出，特征值越大，投影到该特征值对应的特征向量上的方差越大。

因此 PCA 通过保留大部分的样本方差，去掉一些方差小的特征，来实现降低维度。如图2所示，将一个三维空间的数据降到二维，需要选择两个特征值最大的特征向量，相当于把点投影到两个特征向量所在的平面上。

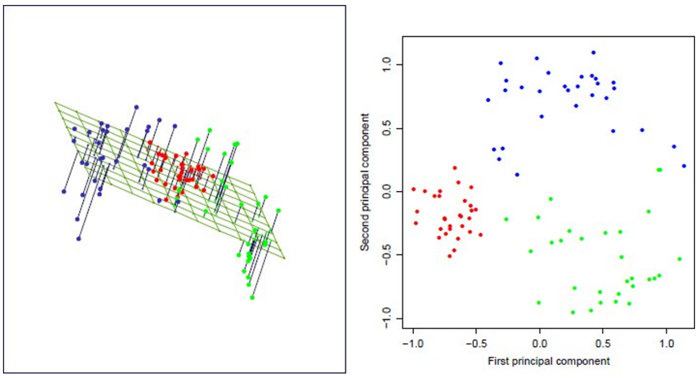

Figure 2: PCA 降维示意图

那么如何确定选择几个主成分？一般采用两种方法：

1. 选择的主成分方差占比之和在高达 $50\% \sim 70\%$，甚至可以更高。一个主成分的方差占比为 $\frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$。
2. 根据特征值的大小来选择，例如选择特征值大于 1 的主成分。

虽然去掉一些主成分确实会丢失一些信息，但因为去掉的主成分特征值较小，并没有损失太多。

3 PCA 的局限性

PCA 在一些特殊情况下会无法使用。比如一些的数据集，数据是由字符串组成。或者是数据很特殊的数据集，例如

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

其协方差矩阵的特征值均为 1，去掉任何一个主成分都是不合适的。

另一种情况是，方差最大的方向并不一定适合分类。如图3所示，若将样本点都投影到方差最大的方向，即纵轴方向的话，两类样本将难以区分。相反，此时最优的方向反而是方差小的方向。

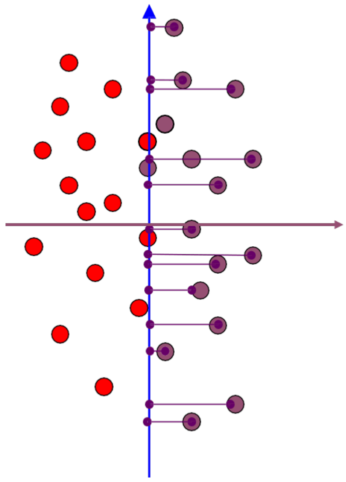

Figure 3: PCA 局限性示意图

4 简单实例

下面用一个简单的实例来说明使用 PCA 的步骤。假设输入的样本矩阵 $X \in \mathbb{R}^{10 \times 2}$，也就是有 10 个样本，每个样本有 2 个特征。

第一步 每个特征减去均值，这样可以简化方差与协方差的运算，而且不影响方差与协方差的值

第二步 计算协方差矩阵

第三步 计算协方差矩阵的特征值与特征向量

第四步 去掉一些不重要主成分

第五步 算出样本在新的坐标系中的坐标，用样本矩阵 X 乘特征向量组成的矩阵 $[v_1, v_2, \ldots, v_k]$

编程实现

```python
import numpy as np

# 数据初始化
X = np.array([[2.5, 2.4],
        [0.5, 0.7],
        [2.2, 2.9],
        [1.9, 2.2],
        [3.1, 3.0],
        [2.3, 2.7],
        [2,   1.6],
        [1,   1.1],
        [1.5, 1.6],
        [1.1, 0.9]])
n, p = X.shape;

# 特征减去均值
X_mean = np.mean(X, axis=0).reshape((1, p))
X = X - np.ones((n, 1)) @ X_mean

# 计算协方差矩阵
```

```python
cov = np.cov(X, rowvar=False)

# 计算特征值与特征向量
eigenvalue, eigenvector = np.linalg.eig(cov)

# 选取主成分
r = np.argsort(-eigenvalue)
v = eigenvector[:, r[0]]

# 转换坐标
X_new = X @ v
print(X_new)

# Output:
# [-0.82797019  1.77758033 -0.99219749 -0.27421042
#   -1.67580142 -0.9129491    0.09910944  1.14457216
#    0.43804614  1.22382056]
```

图4是原样本 X 在二维空间的散点图，图5是 X 在降维后的一维图。

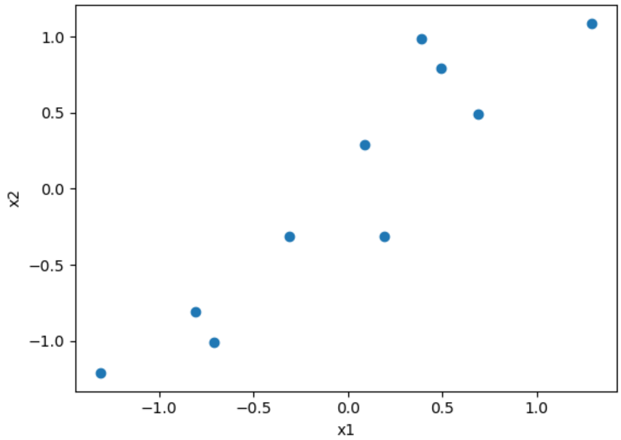

Figure 4: X 的散点图

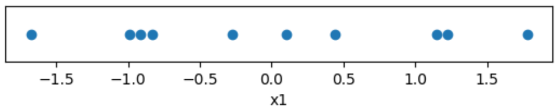

Figure 5: X_new 的散点图

引用

[1] Principal Component Analysis: `https://en.wikipedia.org/wiki/Principal_component_analysis`