无监督学习-降维

ML07

礼欣
www.python123.org

PCA方法及其应用

主成分分析（PCA）

- 主成分分析（Principal Component Analysis，PCA）是最常用的一种降维方法，通常用于高维数据集的探索与可视化，还可以用作数据压缩和预处理等。
- PCA可以把具有相关性的高维变量合成为线性无关的低维变量，称为主成分。主成分能够尽可能保留原始数据的信息。

主成分分析

在介绍PCA的原理之前需要回顾涉及到的相关术语：

- 方差
- 协方差
- 协方差矩阵
- 特征向量和特征值

主成分分析

方差：是各个样本和样本均值的差的平方和的均值，用来度量一组数据的分散程度。

$$\boldsymbol{S^2 = \frac{\sum_{i=1}^{n}(x_i - x)^2}{n-1}}$$

主成分分析

协方差：用于度量两个变量之间的线性相关性程度，若两个变量的协方差为0，则可认为二者线性无关。协方差矩阵则是由变量的协方差值构成的矩阵（对称阵）。

$$\boldsymbol{Cov(X,Y)=\frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{n-1}}$$

主成分分析

特征向量：矩阵的特征向量是描述数据集结构的非零向量，并满足如下公式：

$$\boldsymbol{A\vec{v} = \lambda\vec{v}}$$

A是方阵，$\boldsymbol{\vec{v}}$是特征向量，$\boldsymbol{\lambda}$是特征值。

主成分分析

原理：矩阵的主成分就是其协方差矩阵对应的特征向量，按照对应的特征值大小进行排序，最大的特征值就是第一主成分，其次是第二主成分，以此类推。

主成分分析-算法过程

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

过程：

1: 对所有样本进行中心化: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i$;
2: 计算样本的协方差矩阵 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$;
3: 对协方差矩阵 $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ 做特征值分解;
4: 取最大的 d' 个特征值所对应的特征向量 $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'}$.

输出： 投影矩阵 $\mathbf{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'})$.

图. PCA算法（《机器学习 周志华》）

sklearn中主成分分析

在sklearn库中，可以使用sklearn.decomposition.PCA加载PCA进行降维，主要参数有：

- n_components：指定主成分的个数，即降维后数据的维度
- svd_solver ：设置特征值分解的方法，默认为‘auto’,其他可选有‘full’，‘arpack’，‘randomized’。

PCA实现高维数据可视化

目标：已知鸢尾花数据是4维的，共三类样本。使用PCA实现对鸢尾花数据进行降维，实现在二维平面上的可视化。

萼片长度	萼片宽度	花瓣长度	花瓣宽度	类别
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa
4.3	3	1.1	0.1	Iris-setosa
5.8	4	1.2	0.2	Iris-setosa

图. 鸢尾花数据

实例程序编写

1. 建立工程，导入sklearn相关工具包：

```
>>> import matplotlib.pyplot as plt
#加载matplotlib用于数据的可视化
>>> from sklearn.decomposition import PCA
#加载PCA算法包
>>> from sklearn.datasets import load_iris
#加载鸢尾花数据集导入函数
```

实例程序编写

2. 加载数据并进行降维：

```
>>> data = load_iris()
#以字典形式加载鸢尾花数据集
>>> y = data.target #使用y表示数据集中的标签
>>> X = data.data    #使用X表示数据集中的属性数据
>>> pca = PCA(n_components=2)
#加载PCA算法，设置降维后主成分数目为2
>>> reduced_X = pca.fit_transform(X)
#对原始数据进行降维，保存在reduced_X中
```

实例程序编写

3. 按类别对降维后的数据进行保存：

```
>>> red_x, red_y = [], []
#第一类数据点
>>> blue_x, blue_y = [], []
#第二类数据点
>>> green_x, green_y = [], []
#第三类数据点
```

实例程序编写

3. 按类别对降维后的数据进行保存：

```
for i in range(len(reduced_X)):
    if y[i] == 0:
        red_x.append(reduced_X[i][0])
        red_y.append(reduced_X[i][1])
    elif y[i] == 1:
        blue_x.append(reduced_X[i][0])
        blue_y.append(reduced_X[i][1])
    else:
        green_x.append(reduced_X[i][0])
        green_y.append(reduced_X[i][1])
```

按照鸢尾花的类别将降维后的数据点保存在不同的列表中。

实例程序编写

4. 降维后数据点的可视化：

```
>>> plt.scatter(red_x, red_y, c='r', marker='x')
#第一类数据点
>>> plt.scatter(blue_x, blue_y, c='b', marker='D')
#第二类数据点
>>> plt.scatter(green_x, green_y, c='g', marker='.')
#第三类数据点
>>> plt.show()
#可视化
```

结果展示

可以看出，降维后的数据仍能够清晰地分成三类。这样不仅能削减数据的维度，降低分类任务的工作量，还能保证分类的质量。