



PCA（主程序分析）降维原理小结与实例



王振群

关注她

46 人赞同了该文章

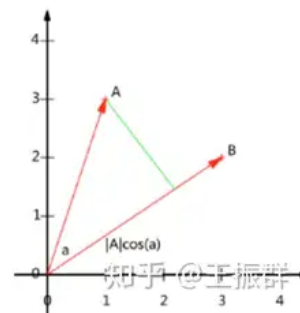
1.主成分分析（Principal components analysis，以下简称PCA）是最重要的降维方法之一。在数据压缩消除冗余和数据噪音消除等领域都有广泛的应用。一般我们提到降维最容易想到的算法就是PCA，目标是基于方差提取最有价值的信息，属于无监督问题。但是降维后的数据因为经过多次矩阵的变化我们不知道降维后的数据意义，但是更加注重降维后的数据结果。

2.向量的表示及基的变换（基：数据的衡量标准，之所以有坐标点（3，2）是因为有x、y轴）：

✎ 内积： $(a_1, a_2, \dots, a_n)^T \cdot (b_1, b_2, \dots, b_n)^T = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$

✎ 解释： $A \cdot B = |A||B|\cos(a)$

✎ 设向量B的模为1，则A与B的内积值等于A向B所在直线投影的矢量长度



基变换：基是正交的（即内积为0或者说相互垂直）

要求：线性无关（x轴和y轴的数据互补影响）

变换：数据与一个基做内积运算，结果作为第一个新的坐标分量，然后与第二个基做内积运算，结果作为第二个新坐标的分量。

▲ 赞同 46 ▼

● 1 条评论

🔗 分享

❤ 喜欢

★ 收藏


📄 申请转载

...



基变换

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{pmatrix} (a_1 \ a_2 \ \cdots \ a_M) = \begin{pmatrix} p_1 a_1 & p_1 a_2 & \cdots & p_1 a_M \\ p_2 a_1 & p_2 a_2 & \cdots & p_2 a_M \\ \vdots & \vdots & \ddots & \vdots \\ p_R a_1 & p_R a_2 & \cdots & p_R a_M \end{pmatrix}$$

 两个矩阵相乘的意义是将右边矩阵中的每一列列向量变换到左边矩阵中每一行行向量为基所表示的空间中去

3.现在已经知道基变换的原理，那么现在问题又落在如何找到一组合适的基呢？

(1) 目标：寻找一个一维基，使得变换为这个基上的坐标表示后，方差最大，也即投影后的投影值尽可能分散。

(2) 方差：描述的是样本集合的各个样本点到均值的距离之平均，一般用来描述一维数据的

在概率论和统计方差用来度量随机变量和数据期望之间的偏离程度。

方差：

$$Var(a) = \frac{1}{m} \sum_{i=1}^m (a_i - \mu)^2$$

协方差：是一种用来度量两个随机变量关系的统计量，只能处理二维问题。

协方差（假设均值为0时）：

$$Cov(a, b) = \frac{1}{m} \sum_{i=1}^m a_i b_i$$

方差和协方差的关系：方差是用来度量单个变量的“自身变异大小”的总体参数，方差越大表面该变量变异越大。协方差是两个变量相互影响大小的参数，协方差的参数越大，则两个变量相互影响越大。

这时引入协方差。如果协方差为0，表示两个特征是线性无关的，所以为了尽可能的展示数据更多的原始信息，我们需要特征之间是线性无关的，也即协方差为0。**第一个坐标轴选择方差最大的，为了是协方差为0，选择第二个基时只能在第一个基正交的平面上选择。**

4.优化目标：将一组N维向量降为K（ $0 < k < n$ ）维，目标选择K个单位正交基，使得原始数据变换到这组基上后，各字段两两间协方差为0，字段的方差尽可能最大。

$$\text{协方差矩阵: } X = \begin{pmatrix} a_1 & a_2 & \cdots & a_m \\ b_1 & b_2 & \cdots & b_m \end{pmatrix} \quad \frac{1}{m} X X^T = \begin{pmatrix} \frac{1}{m} \sum_{i=1}^m a_i^2 & \frac{1}{m} \sum_{i=1}^m a_i b_i \\ \frac{1}{m} \sum_{i=1}^m a_i b_i & \frac{1}{m} \sum_{i=1}^m b_i^2 \end{pmatrix}$$

协方差矩阵对角线上的元素分别是字段的方差，而其他元素是a和b的协方差。并且是实对称矩阵。

实对称矩阵：一个n行n列的实对称矩阵一定可以找到n个单位正交特征向量

$$E = (e_1 \quad e_2 \quad \cdots \quad e_n)$$

实对称阵可进行对角化：

$$E^T C E = \Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

根据特征值的从大到小，将特征向量从上到下排列，则用前K行组成的矩阵乘以原始数据矩阵X，就得到了我们需要的降维后的数据矩阵Y。

到此求出特征值和特征向量，并由此求出最后的解。

5.PCA实例：

PCA实例

数据： $\begin{pmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{pmatrix}$

协方差矩阵： $C = \frac{1}{5} \begin{pmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 \\ -1 & 0 \\ 0 & 0 \\ 2 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{6}{5} & \frac{4}{5} \\ \frac{4}{5} & \frac{6}{5} \end{pmatrix}$

特征值： $\lambda_1 = 2, \lambda_2 = 2/5$ 特征向量： $c_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix}, c_2 \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

对角化： $PCP^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 6/5 & 4/5 \\ 4/5 & 6/5 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2/5 \end{pmatrix}$

降维： $Y = (1/\sqrt{2} \quad 1/\sqrt{2}) \begin{pmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{pmatrix} = (-3/\sqrt{2} \quad -1/\sqrt{2} \quad 0 \quad 3/\sqrt{2} \quad 1/\sqrt{2})$

6. PCA算法总结

这里对PCA算法做一个总结。作为一个非监督学习的降维方法，它只需要特征值分解，就可以对数据进行压缩，去噪。因此在实际场景应用很广泛。为了克服PCA的一些缺点，出现了很多PCA的变种，比如第六节的为解决非线性降维的KPCA，还有解决内存限制的增量PCA方法Incremental PCA，以及解决稀疏数据降维的PCA方法Sparse PCA等。

PCA算法的主要优点有：

- 1) 仅仅需要以方差衡量信息量，不受数据集以外的因素影响。
- 2) 各主成分之间正交，可消除原始数据成分间的相互影响的因素。
- 3) 计算方法简单，主要运算是特征值分解，易于实现。

PCA算法的主要缺点有：

- 1) 主成分各个特征维度的含义具有一定的模糊性，不如原始样本特征的解释性强。
- 2) 方差小的非主成分也可能含有对样本差异的重要信息，因降维丢弃可能对后续数据处理有影响。

二、基于sklearn实现PCA降维

下面我们主要基于sklearn.decomposition.PCA来讲解如何使用scikit-learn进行PCA降维。PCA类基本不需要调参，一般来说，我们只需要指定我们需要降维到的维度，或者我们希望降维后的主成分的方差和占原始维度所有特征方差和的比例阈值就可以了。

现在我们对sklearn.decomposition.PCA的主要参数做一个介绍：

1) **n_components**：这个参数可以帮我们指定希望PCA降维后的特征维度数目。最常用的做法是直接指定降维到的维度数目，此时n_components是一个大于等于1的整数。当然，我们也可以指定主成分的方差和所占的最小比例阈值，让PCA类自己去根据样本特征方差来决定降维到的维度数，此时n_components是一个 (0, 1]之间的数。当然，我们还可以将参数设置为"mle"，此时PCA类会用MLE算法根据特征的方差分布情况自己去选择一定数量的主成分特征来降维。我们也可以用默认值，即不输入n_components，此时n_components=min(样本数, 特征数)。

2) **whiten**：判断是否进行白化。所谓白化，就是对降维后的数据的每个特征进行归一化，让方差都为1.对于PCA降维本身来说，一般不需要白化。如果你PCA降维后有后续的数据处理动作，可以考虑白化。默认值是False，即不进行白化。

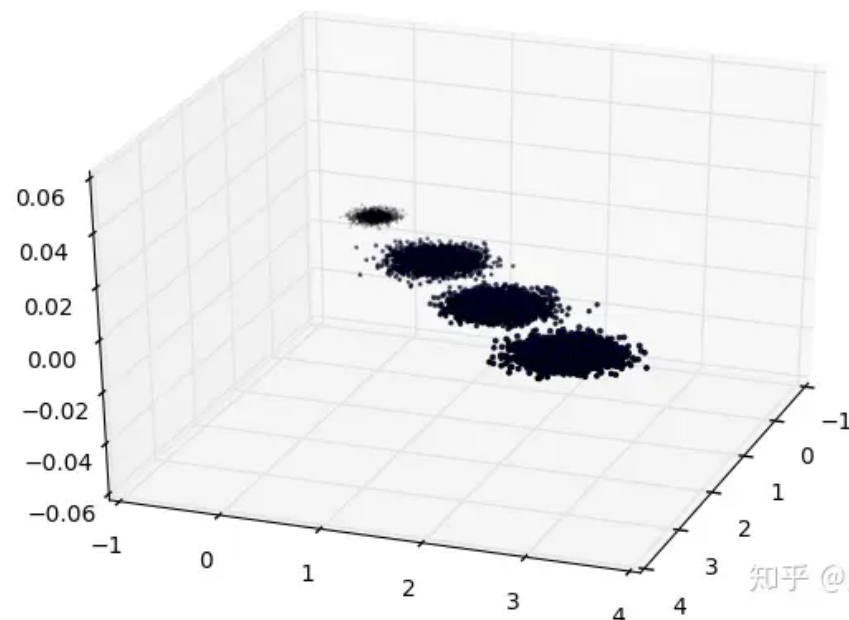
3) **svd_solver**：即指定奇异值分解SVD的方法，由于特征分解是奇异值分解SVD的一个特例，一般的PCA库都是基于SVD实现的。有4个可以选择的值：{ 'auto' , 'full' , 'arpack' , 'randomized' }。randomized一般适用于数据量大，数据维度多同时主成分数目比例又较低的PCA降维，它使用了一些加快SVD的随机算法。full则是传统意义上的SVD，使用了scipy库对应的实现。arpack和randomized的适用场景类似，区别是randomized使用的是scikit-learn自己的SVD实现，而arpack直接使用了scipy库的sparse SVD实现。默认是auto，即PCA类会自己去在前面讲到的三种算法里面去权衡，选择一个合适的SVD算法来降维。一般来说，使用默认值就够了。

除了这些输入参数外，有两个PCA类的成员值得关注。第一个是**explained_variance_**，它代表降维后的各主成分的方差值。方差值越大，则说明越是重要的主成分。第二个是**explained_variance_ratio_**，它代表降维后的各主成分的方差值占总方差值的比例，这个比例越大，则越是重要的主成分。

1.首先生成随机数据并可视化

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
from sklearn.datasets.samples_generator import make_blobs
# X为样本特征, Y为样本簇类别, 共10000个样本, 每个样本3个特征, 共4个簇
X, y = make_blobs(n_samples=10000, n_features=3, centers=[[3,3, 3], [0,0,0], [1,1,1],
[2,2,2]], cluster_std=[0.2, 0.1, 0.2, 0.2],
                  random_state=9)
fig = plt.figure()
ax = Axes3D(fig, rect=[0, 0, 1, 1], elev=30, azim=20)
plt.scatter(X[:, 0], X[:, 1], X[:, 2], marker='o')
```

知乎 @王振群



2.我们先不降维, 只对数据进行投影, 看看投影后的三个维度的方差分布, 代码如下:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
pca.fit(X)
print pca.explained_variance_ratio_
print pca.explained_variance_
```

输出如下：

```
[ 0.98318212  0.00850037  0.00831751]
[ 3.78483785  0.03272285  0.03201892]
```

可以看出投影后三个特征维度的方差比例大约为98.3% : 0.8% : 0.8%。投影后第一个特征占了绝大多数的主成分比例。

3.我们将数据从三维降到二维

```
pca = PCA(n_components=2)
pca.fit(X)
print pca.explained_variance_ratio_
print pca.explained_variance_
```

输出如下：

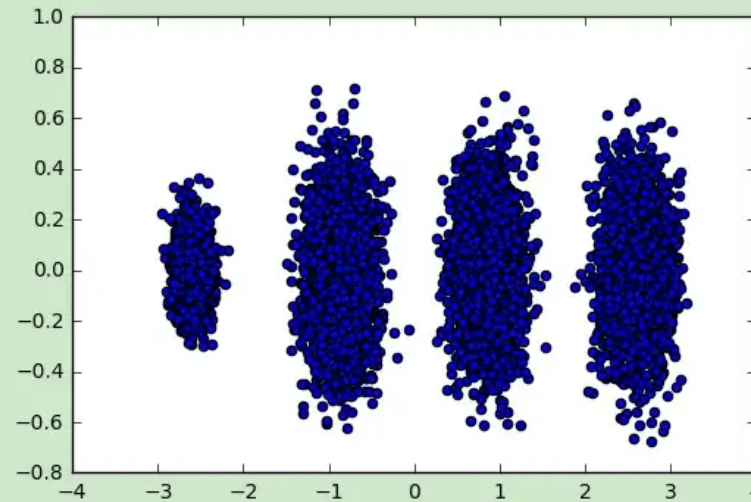
```
[ 0.98318212  0.00850037]
[ 3.78483785  0.03272285]
```

这个结果其实可以预料，因为上面三个投影后的特征维度的方差分别为：[3.78483785 0.03272285 0.03201892]，投影到二维后选择的肯定是前两个特征，而抛弃第三个特征

为了有个直观的认识，我们看看此时转化后的数据分布，代码如下：

```
X_new = pca.transform(X)
plt.scatter(X_new[:, 0], X_new[:, 1], marker='o')
plt.show()
```

输出的图如下：



可见降维后的数据依然可以很清楚的看到我们之前三维图中的4个簇。

知乎 @王振群

3.现在我们看看不直接指定降维的维度，而指定降维后的主成分方差和比例

```
pca = PCA(n_components=0.95)
pca.fit(X)
print pca.explained_variance_ratio_
print pca.explained_variance_
print pca.n_components_
```

我们指定了主成分至少占95%，输出如下：

```
[ 0.98318212]
[ 3.78483785]
1
```

可见只有第一个投影特征被保留。这也很好理解，我们的第一个主成分占投影特征的方差比例高达98%。

知乎 @王振群

4.最后我们看看PCA算法自己选择降维维度的结果


```
pca = PCA(n_components='mle')
pca.fit(X)
print pca.explained_variance_ratio_
print pca.explained_variance_
print pca.n_components_
```

输出结果如下：

```
[ 0.98318212]
[ 3.78483785]
1
```

可见由于我们的数据的第一个投影特征的方差占比高达98.3%，MLE算法只保留了我们的第一个特征。

参考：[刘建平博客园](#)

发布于 2019-02-18 22:51

机器学习 降维算法



理性发言，友善互动

1 条评论

默认 最新



Kevin

看了那么多PCA原理，只有你这个是跟我老师讲的一样的😄

2019-03-06

回复 2

文章被以下专栏收录



小绕搞数据分析

一个慢慢头变秃的地方

推荐阅读

特征工程系列之降维：用PCA压缩数据

引言降维是关于摆脱“无信息的信息”的同时保留关键点。有很多方法可以定义“无信息”。PCA 侧重于线性依赖的概念。我们将数据矩阵的列空间描述为所有特征向量的跨度。如果列空间与特征的总...

墨明棋妙 发表于特征工程系...



常见是数据降维方法小结--PCA,ICA,SVD,FA

马卡斯·扬 发表于病理图像处...



降维方法小结和理解：PCA、LDA、MDS、ISOMAP、...

远行人

主成分分析降维原理——PCA数学推导

1.PCA降维的计算过程 下图是从西瓜书里截取的PCA降维过程的图片。需要说明的是，算法中的向量为列向量。假设原始维度为d，样本数目为m，因此特征矩阵X的维度为d×m，W的维度为d×d'。降维...

南瓜派三蔬

译