**scikit-learn——特征降维**

**任务目标**

1.了解数据降维的各种算法原理

2.熟练掌握sklearn.decomposition中降维方法的使用

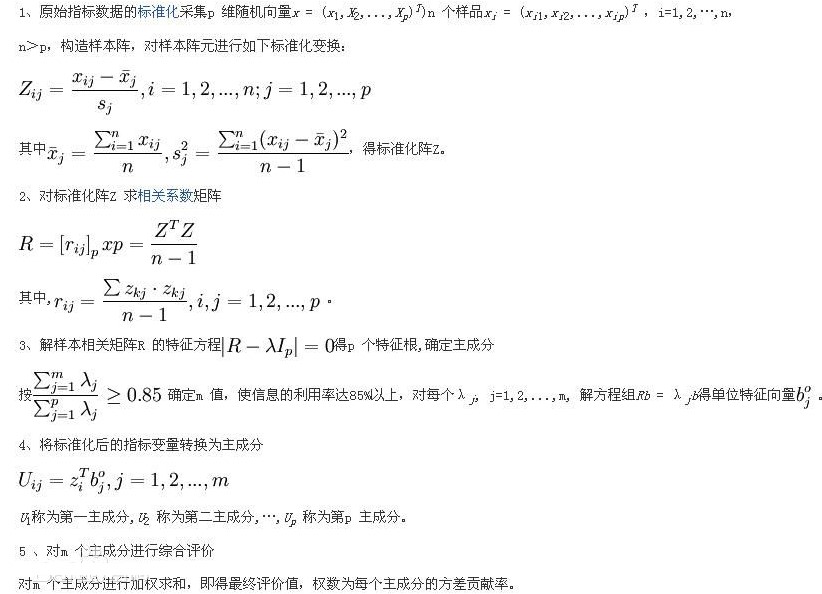
**相关知识**

sklearn.decomposition

主成分分析算法（Principal Component Analysis， PCA）的目的是找到能用较少信息描述数据集的特征组合。它意在发现彼此之间没有相关性、能够描述数据集的特征，确切说这些特征的方差跟整体方差没有多大差距，这样的特征也被称为主成分。这也就意味着，借助这种方法，就能通过更少的特征捕获到数据集的大部分信息。

主成分分析原理

设法将原来变量重新组合成一组新的相互无关的几个综合变量，同时根据实际需要从中可以取出几个较少的总和变量尽可能多地反映原来变量的信息的统计方法叫做主成分分析或称主分量分析，也是数学上处理降维的一种方法。主成分分析是设法将原来众多具有一定相关性（比如P个指标），重新组合成一组新的互相无关的综合指标来代替原来的指标。通常数学上的处理就是将原来P个指标作线性组合，作为新的综合指标。最经典的做法就是用F1（选取的第一个线性组合，即第一个综合指标）的方差来表达，即Va（rF1）越大，表示F1包含的信息越多。因此在所有的线性组合中选取的F1应该是方差最大的，故称F1为第一主成分。如果第一主成分不足以代表原来P个指标的信息，再考虑选取F2即选第二个线性组合，为了有效地反映原来信息，F1已有的信息就不需要再出现在F2中，用数学语言表达就是要求Cov（F1,F2）=0，则称F2为第二主成分，依此类推可以构造出第三、第四，……，第P个主成分。



sklearn中主成分分析的模型

class sklearn.decomposition.PCA(n\_components=None, copy=True, whiten=False, svd\_solver=’auto’, tol=0.0, iterated\_power=’auto’, random\_state=None）

sklearn.decomposition.PCA参数介绍

接下来主要基于sklearn.decomposition.PCA类来讲解如何使用scikit-learn进行PCA降维。PCA类基本不需要调参，一般来说，只需要指定要降维到的维度，或者希望降维后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。

现在介绍一下sklearn.decomposition.PCA的主要参数：

1) n\_components：这个参数指定了希望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，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]之间的浮点数。当然，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。也可以使用默认值，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。如果在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库中的实现。'arpack'和'randomized'的适用场景类似，区别是'randomized'使用的是scikit-learn中的SVD实现，而'arpack'直接使用了scipy库的sparse SVD实现。默认是'auto'，即PCA类会自己去权衡前面讲到的三种算法，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。

**任务内容**

练习scikit-learn中特征降维的函数或方法。

**系统环境**

Python

Jupyter

**任务步骤**

1.打开终端模拟器，在命令行输入ipython notebook --ip='127.0.0.1

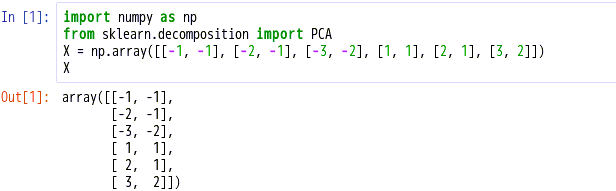
2.新建一个ipynb文件，用于编写并执行代码。

3.导入numpy模块，别名为np，从sklearn.decomposition中导入PCA类。

1. **import** numpy as np
2. from sklearn.decomposition **import** PCA

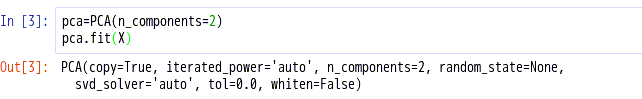
4.使用numpy的array函数创建一个二维数组X。

1. X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
2. X



5.使用PCA类创建一个主成分分析的对象pca，设置PCA算法中所要保留的主成分个数参数为n\_cimponents=2，调用pca对象中的fit方法用数据X来训练PCA模型，函数返回值是调用fit方法的对象pca本身。

1. pca=PCA(n\_components=2)
2. pca.fit(X)



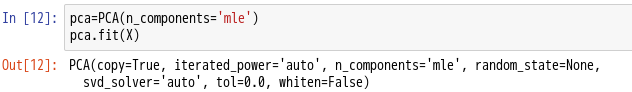
6.打印所保留的2个成分各自的方差百分比。

1. pca.explained\_variance\_ratio\_



7.使用PCA类创建一个主成分分析的对象pca，设置使用MLE算法计算降维后维度数量，调用pca对象中的fit方法用数据X来训练PCA模型，函数返回值是调用fit方法的对象pca本身。

1. pca=PCA(n\_components='mle')
2. pca.fit(X)



8.打印各维度的方差

1. pca.explained\_variance\_



9.各维度的方差值占总方差值的比例

1. pca.explained\_variance\_ratio\_



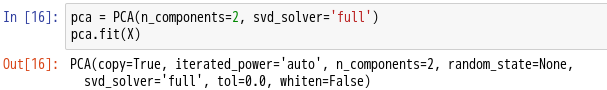
10.降维后的维度数量

1. pca.n\_components\_



11.使用PCA类创建一个主成分分析的对象pca，设置PCA算法中所要保留的主成分个数参数为n\_components=2，指定奇异值分解SVD的方法参数svd\_solver='full'，调用pca对象中的fit方法用数据X来训练PCA模型，函数返回值是调用fit方法的对象pca本身。

1. pca = PCA(n\_components=2, svd\_solver='full')
2. pca.fit(X)



12.打印所保留的2个成分各自的方差百分比。

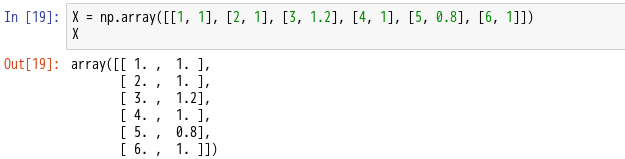
1. pca.explained\_variance\_ratio\_



非负矩阵因子分解 (NMF)

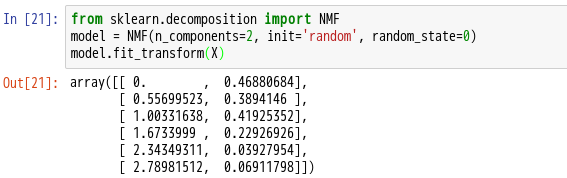
13.使用numpy的array函数创建一个二维数组X。

1. X = np.array([[1, 1], [2, 1], [3, 1.2], [4, 1], [5, 0.8], [6, 1]])
2. X



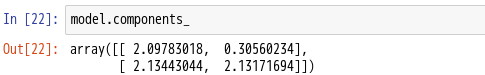
14.导入sklearn.decomposition的NMF类，使用NMF类创建一个非负矩阵因子分解实例对象，使用fit\_transform方法训练模型并用模型对数据X进行非负矩阵因子分解，

1. from sklearn.decomposition **import** NMF
2. model = NMF(n\_components=2, init='random', random\_state=0)
3. model.fit\_transform(X)



15.最后查看分解后数据的因子数量。

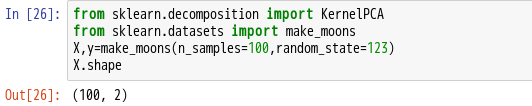
1. model.components\_



核主成分分析

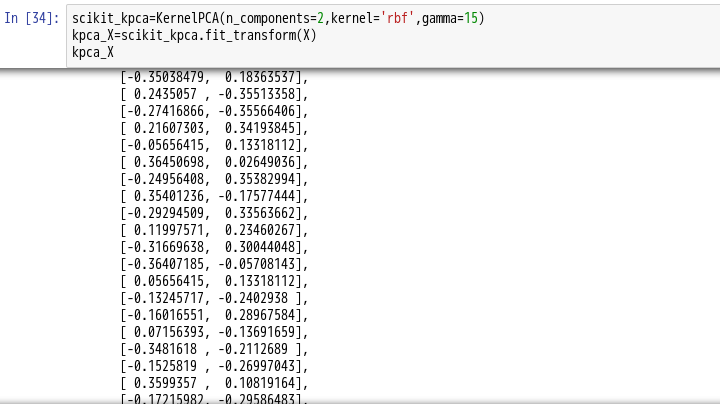
16.导入sklearn.decomposition中的KernelPCA类，sklearn.datasets中的make\_moons，使用make\_moons函数创建样本数为50的训练数据X，y。

1. from sklearn.decomposition **import** KernelPCA
2. from sklearn.datasets **import** make\_moons
3. X,y=make\_moons(n\_samples=100,random\_state=123)
4. X.shape



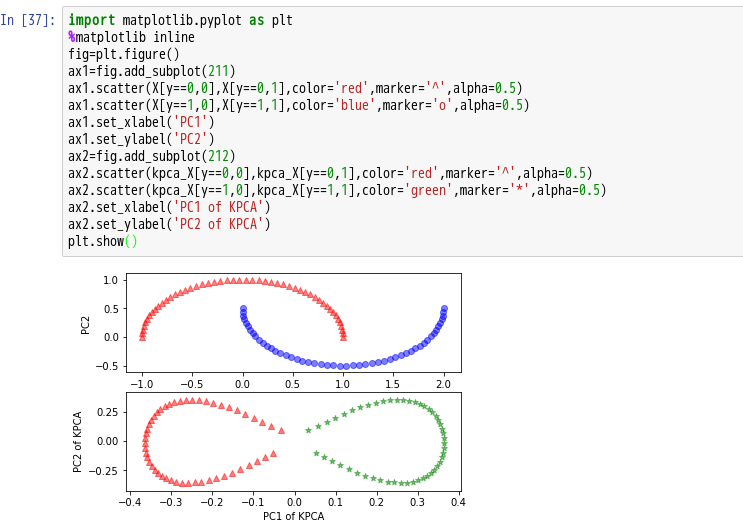
17.使用KernelPCA类创建一个核主成分分析的对象scikit\_kpca，设置要保留的主成分个数参数n\_components=2，核函数使用径向基核函数kernel=’rbf'，内核系数参数gamma=15，然后使用对象的fit\_transform方法通过X数据训练模型，并用模型对X数据进行核主成分分析。

1. scikit\_kpca = KernelPCA(n\_components=2,kernel='rbf',gamma=15)
2. kpca\_X = scikit\_kpca.fit\_transform(X)
3. kpca\_X



18.导入matplotlib.pyplot模块，别名为plt，使用plt绘制训练数据X,y的散点图，与kpca\_X,y核主成分分析返回数据的散点图，代码如下。

1. **import** matplotlib.pyplot as plt
2. %matplotlib inline
3. fig=plt.figure()
4. ax1=fig.add\_subplot(211)
5. ax1.scatter(X[y==0,0],X[y==0,1],color='red',marker='^',alpha=0.5)
6. ax1.scatter(X[y==1,0],X[y==1,1],color='blue',marker='o',alpha=0.5)
7. ax1.set\_xlabel('PC1')
8. ax1.set\_ylabel('PC2')
9. ax2=fig.add\_subplot(212)
10. ax2.scatter(kpca\_X[y==0,0],kpca\_X[y==0,1],color='red',marker='^',alpha=0.5)
11. ax2.scatter(kpca\_X[y==1,0],kpca\_X[y==1,1],color='green',marker='\*',alpha=0.5)
12. ax2.set\_xlabel('PC1 of KPCA')
13. ax2.set\_ylabel('PC2 of KPCA')
14. plt.show()



以上！