主成分分析(PCA)在鸢尾花数据集(Iris)上的应用

一、什么是PCA

PCA是一种用于提取数据集中的模式的统计技术，它做的就是转换数据集以识别隐藏的关系，相似性或差异，然后可以在其输出上进行降维，数据压缩或特征提取。不过它更为人所熟知的是降维。

那么我们为什么要降维，降维不是意味着数据信息的丢失吗？此言不假，但是在机器学习领域，多数情况下我们遇到的数据集都是有大量特征的，动辄十几条甚至是几十条特征。这种情况下，如果不做处理就直接将数据"喂给"机器学习算法的话，很有可能会造成维度灾难。这样机器学习算法就会失效。所以，降维技术就进入了人们的视线。这篇文章将会来谈一谈PCA算法的工作流程以及基于Python的简单实现，涉及到数学证明的部分很少。

二、PCA的工作流程

PCA的基本流程可以分为下面五个步骤：

> 1. 读入数据
> 2. 计算数据的协方差矩阵(covariance matrix of data)
> 3. 计算协方差矩阵上的特征值和特征向量(eigenvalues and eigenvectors)
> 4. 选择主要成分(principal components)
> 5. 从所选成分构造新的特征数据集

三、关于鸢尾花(Iris)数据集

本次实验将要在著名的鸢尾花(Iris)数据集上进行测试，如果对于这个数据集比较陌生，可以看看这个教程，对于鸢尾花数据集的剖析不可谓不深刻。不过我还是简单说说这个数据集，如图所示

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

这个数据集有1个标签Species和4个特征(花的类型)。所以，这个数据是四维数据。虽然维度不算很大，但是我们今天还是要尝试给他降维。

四、使用Python进行实验

1. **导入数据**

```
from sklearn import datasets
import numpy as np

iris = datasets.load_iris()
R = np.array(iris.data)
```

这样，我们就将数据加载到一个名为R的矩阵中，共有150个样本(x轴)和4个特征(y轴)。

2. **计算协方差矩阵**

协方差矩阵就是特征(维)的协方差矩阵。协方差是两个特征的方差；换句话说，两个特性之间的差异。当需要从现有特性中提取新的模式或特性时，这是非常有用的信息。因此我们需要计算数据集的协方差矩阵。由于数据中有4个特征，我们需要计算6个协方差和4个方差。

```
R_cov = np.cov(R, rowvar=False)

import pandas as pd
iris_covmat = pd.DataFrame(data=R_cov, columns=iris.feature_names)
iris_covmat.index = iris.feature_names
```

下面是结果协方差矩阵。除了对角线上的元素外，其他元素对于PCA是必不可少的。

当cov(a, b) = cov(b, a)时，协方差矩阵对角线的上、下边长相等。如果(a, b)的协方差为正，则a和b同时变化;如果是负的，它们变化方向相反。例如，在下面的矩阵中，数据集中一朵花的petal_length和petal_width特征具有正协方差(1.296287)，表示这两个特征同时增加或减少。

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	0.685694	-0.039268	1.273682	0.516904
sepal width (cm)	-0.039268	0.188004	-0.321713	-0.117981
petal length (cm)	1.273682	-0.321713	3.113179	1.296387
petal width (cm)	0.516904	-0.117981	1.296387	0.582414

3. **计算特征值和特征向量**

特征值和特征向量是PCA的核心；不仅是PCA，还有SVD，LDA。

有三点需要注意一下：第一，我们只能计算一个方阵的特征值/特征向量(n x n，矩阵的协方差)。第二，特征向量互相正交。如果我们有n维矩阵那么我们有n个特征向量在n空间中它们都是正交的的。这是有意义的，因为它们都构成了它们所表示的数据。最后，每个特征值对应一个特征值。

由于我们在寻找新的特征来降低数据的维数，因此计算数据协方差矩阵的特征向量来寻找具有显著性(特征值)的模式(特征向量)。协方差矩阵的特征向量就表示新的特征。

```
eig_values, eig_vectors = np.linalg.eig(R_cov)
```

```
eig_values
```

```
array([4.22484077, 0.24224357, 0.07852391, 0.02368303])
```

```
eig_vectors
```

```
array([[ 0.36158968, -0.65653988, -0.58099728,  0.31725455],
       [-0.08226889, -0.72971237,  0.59641809, -0.32409435],
       [ 0.85657211,  0.1757674 ,  0.07252408, -0.47971899],
       [ 0.35884393,  0.07470647,  0.54906091,  0.75112056]])
```

4. **选择主要成分**

从第一个结果中，我们得到了数据中的特征值和相应的特征向量，如上所示。我们需要做的是把特征值从高到低排序。然后，我们选择一些值最大的特征向量来构建我们的新特征。因此我们必须选择特征值较高的特征向量。在这种情况下，如果我们想要将Iris数据的维数减少到2，我们将选择前两个特征向量，并把这个矩阵称为新特征向量的矩阵。

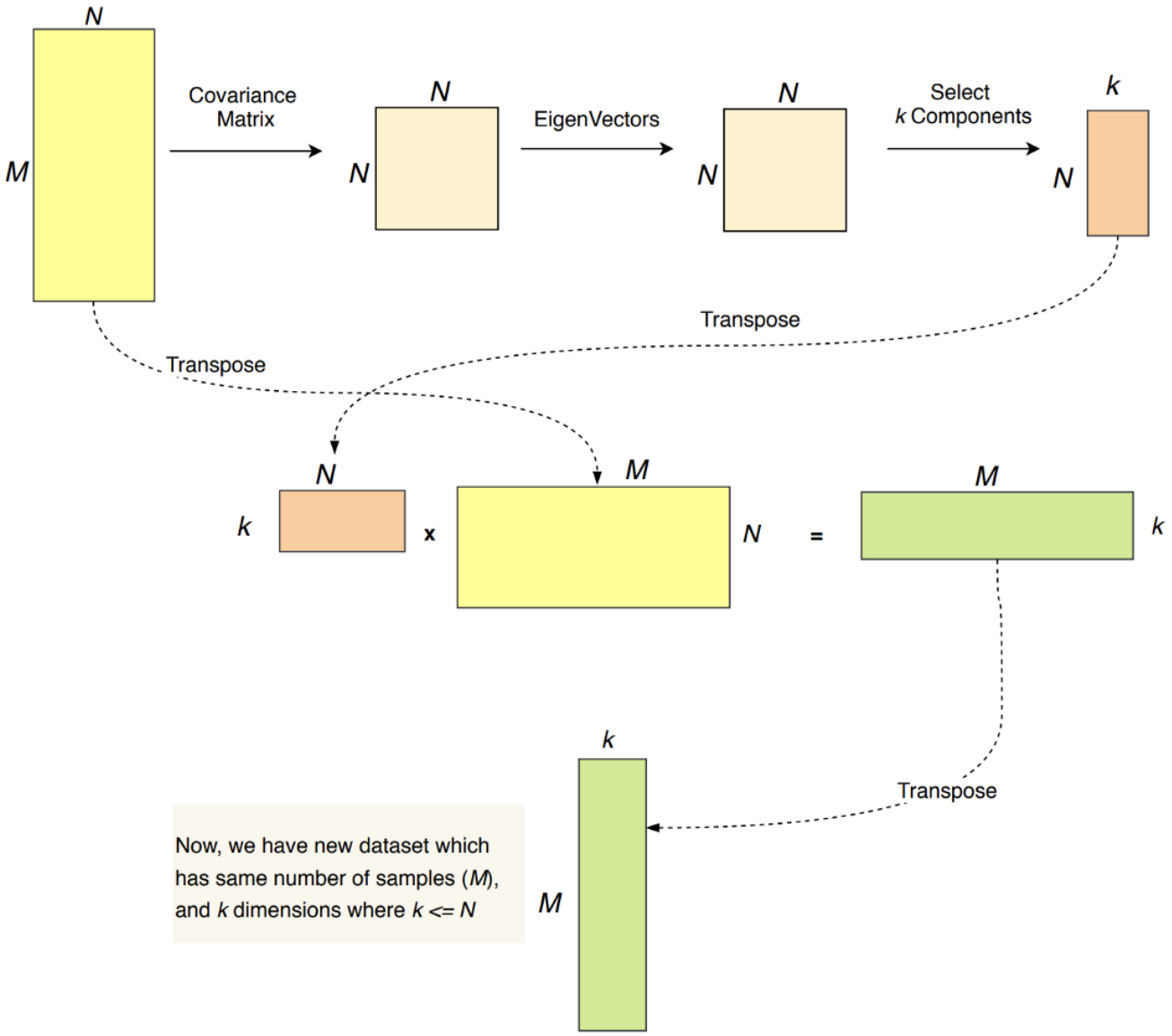
```
featureVector = eig_vectors[:,:2]
```

通过去掉一些分量或者特征值/特征向量我们会丢失一些信息。但我们拥有了更少维度的数据。

5. **构建新的数据**

现在，为了建立新的数据集，我们需要将新特征向量(选定的主成分)的转置左乘原始矩阵(R)的转置。

$$newDataset^T = featureVector^T * dataset^T$$



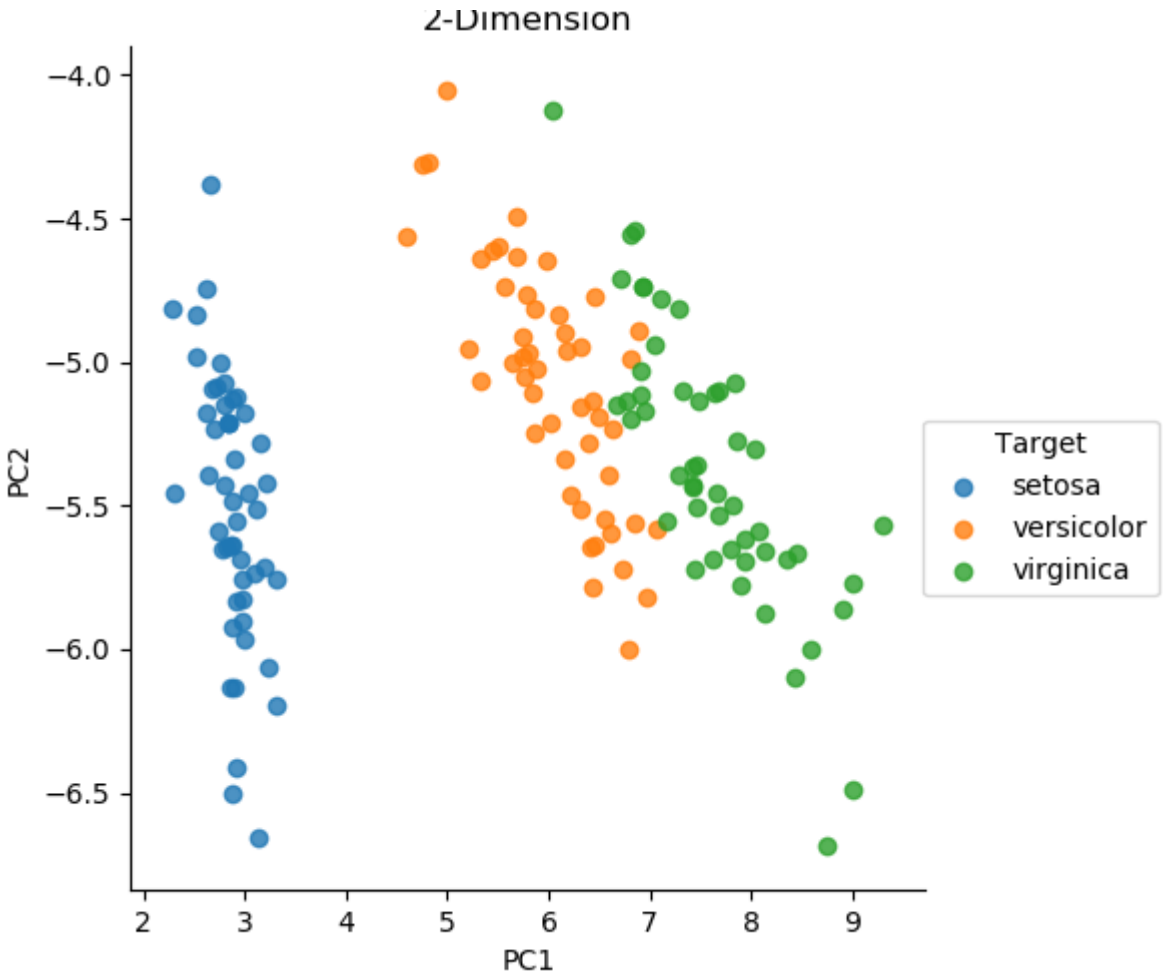

```
featureVector_t = np.transpose(featureVector)

# R is the original iris dataset
R_t = np.transpose(R)

newDataset_t = np.matmul(featureVector_t, R_t)
newDataset = np.transpose(newDataset_t)
```

我们便得到了二维简化的新数据集。

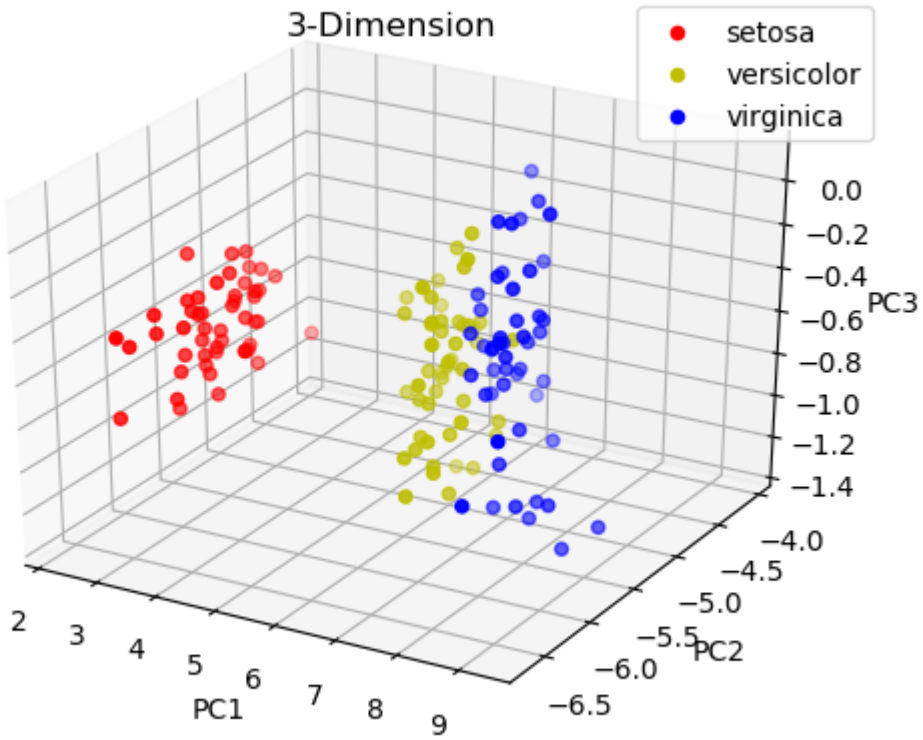
6. **可视化**

```
import seaborn as sns
import pandas as pd
%matplotlib inline

# create new DataFrame
df = pd.DataFrame(data=newDataset, columns=['PC1', 'PC2'])
y = pd.Series(iris.target)
```

```
y = y.replace(0, 'setosa')
y = y.replace(1, 'versicolor')
y = y.replace(2, 'virginica')
df['Target'] = y

# plot 2D data
sns.lmplot(x='PC1', y='PC2', data=df, hue='Target', fit_reg=False, legend=True)
```



如果想要降维到3维，也很简单，在这里就不做过多介绍了，放一张图。



五、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：

- 仅仅需要以方差衡量信息量，不受数据集以外的因素影响。
- 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。
- 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：

- 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。
- 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。