主成分分析（PCA）

目录

学习目标

- 理解PCA分析的原理
- 理解PCA的分析过程
- 能在实际项目中使用PCA分析来解决复杂问题

PCA 主成分分析的通俗讲解

1 什么是降维?

比如说有如下的房价数据：

	房价(百万元)
a	10
b	2
c	1
d	7
e	3

这种一维数据可以直接放在实数轴上：

不过数据还需要处理下，假设房价样本用X表示，那么均值为：

$$\overline{X} = \frac{X_1 + X_2 + X_3 + X_4 + X_5}{5} = \frac{10 + 2 + 1 + 7 + 3}{5} = 4.6$$

然后以均值$\overline{X}$为原点：

以$\overline{X}$为原点的意思是，以$\overline{X}$为0，那么上述表格的数字就需要修改下：

	房价(百万元)
a	$10 - \overline{X} = 5.4$
b	$2 - \overline{X} = -2.6$
c	$1 - \overline{X} = -3.6$
d	$7 - \overline{X} = 2.4$
e	$3 - \overline{X} = -1.6$

这个过程称为“中心化”。“中心化”处理的原因是，这些数字后继会参与统计运算，比如求样本方差，中间就包含了$X_i - \overline{X}$：

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2$$

这里说明一下，虽然样本方差的分母是应该为n-1，这里分母采用n是因为这样算出来的样本方差$Var(X)$为一致估计量，不会太影响计算结果并且可以减小运算负担。

用“中心化”的数据就可以直接算出“房价”的样本方差：

$$Var(X) = \frac{1}{n}\left(5.4^2 + (-2.6)^2 + (-3.6)^2 + 2.4^2 + (-1.6)^2\right)$$

“中心化”之后可以看出数据大概可以分为两类：

现在新采集了房屋的面积，可以看出两者完全正相关，有一列其实是多余的：

	房价(百万元)	面积(百平米)
a	10	10
b	2	2
c	1	1
d	7	7
e	3	3

求出房屋样本、面积样本的均值，分别对房屋样本、面积样本进行“中心化”后得到：

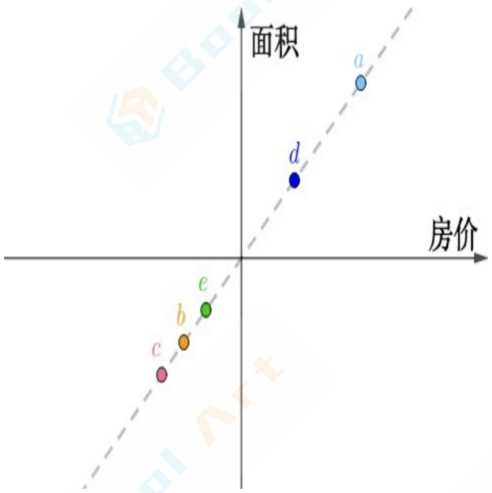
	房价(百万元)	面积(百平米)
a	5.4	5.4
b	−2.6	−2.6
c	−3.6	−3.6
d	2.4	2.4
e	−1.6	−1.6

房价(X)和面积(Y)的样本协方差是这样的（这里也是用的一致估计量）：

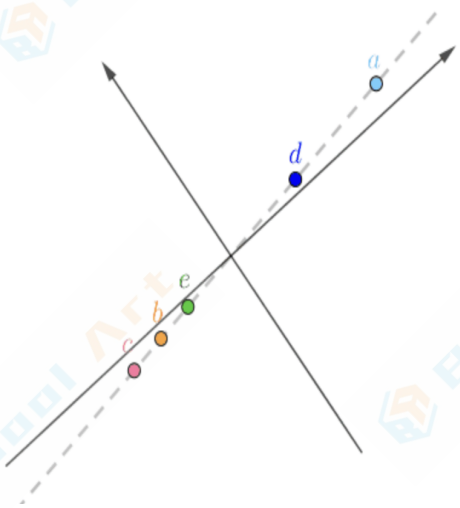
$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})$$

可见“中心化”后的数据可以简化上面这个公式，这点后面还会看到具体应用。

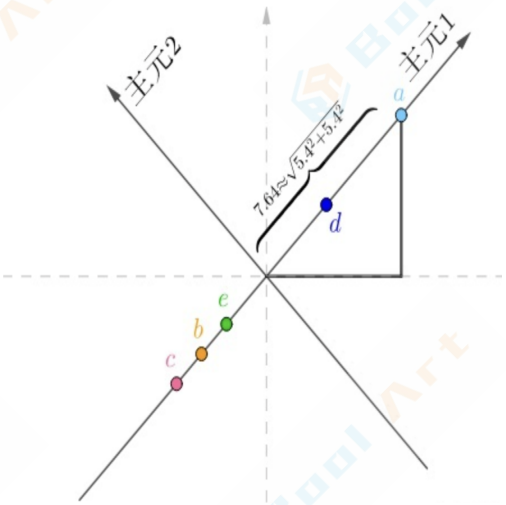
把这个二维数据画在坐标轴上，横纵坐标分别为“房价”、“面积”，可以看出它们排列为一条直线：



如果旋转坐标系，让横坐标和这条直线重合：



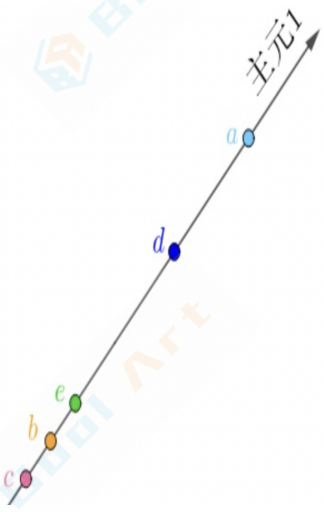
旋转后的坐标系，横纵坐标不再代表“房价”、“面积”了，而是两者的混合（术语是线性组合），这里把它们称作“主元1”、“主元2”，坐标值很容易用勾股定理计算出来，比如a在“主元1”的坐标值为：



很显然a在“主元2”上的坐标为0，把所有的房间换算到新的坐标系上：

	主元1	主元2
a	7.64	0
b	−3.68	0
c	−5.09	0
d	3.39	0
e	−2.26	0

因为“主元2”全都为0，完全是多余的，我们只需要“主元1”就够了，这样就又把数据降为了一维，而且没有丢失任何信息：



2 非理想情况如何降维?

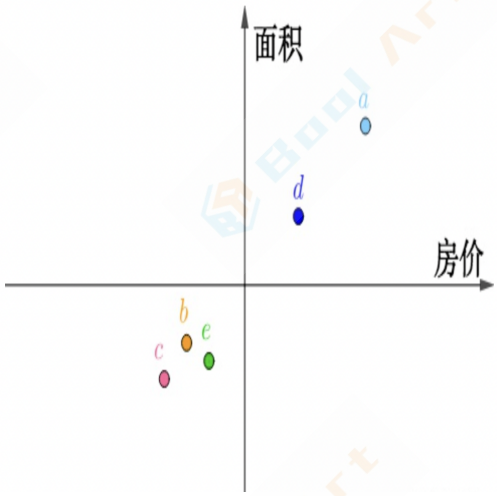
上面是比较极端的情况，就是房价和面积完全正比，所以二维数据会在一条直线上。 现实中虽然正比，但总会有些出入：

	房价(百万元)	面积(百平米)
a	10	9
b	2	3
c	1	2
d	7	6.5
e	3	2.5

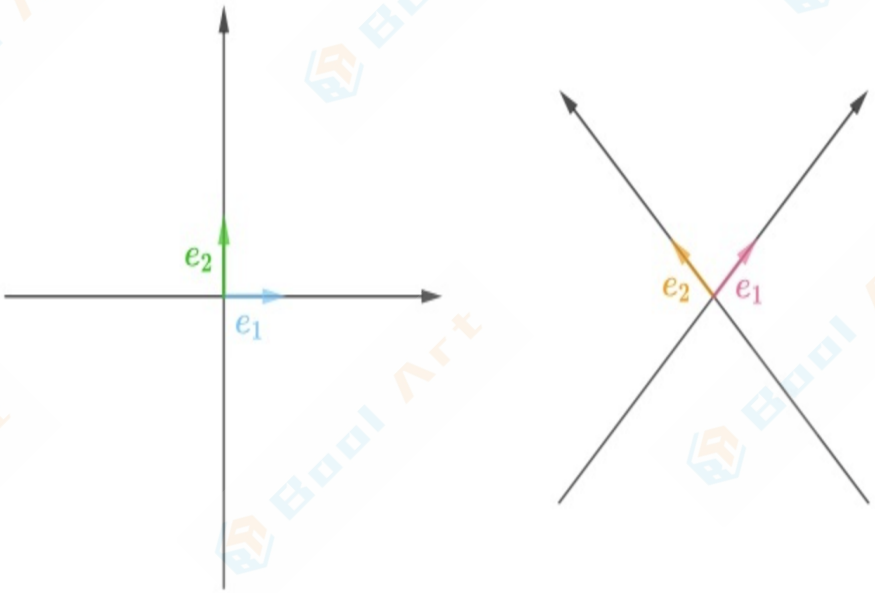
中心化 →

	房价(百万元)	面积(百平米)
a	5.4	4.4
b	-2.6	-1.6
c	-3.6	-2.6
d	2.4	1.9
e	-1.6	-2.1

把这个二维数据画在坐标轴上，横纵坐标分别为“房价”、“面积”，虽然数据看起来很接近一条直线，但是终究不在一条直线上：



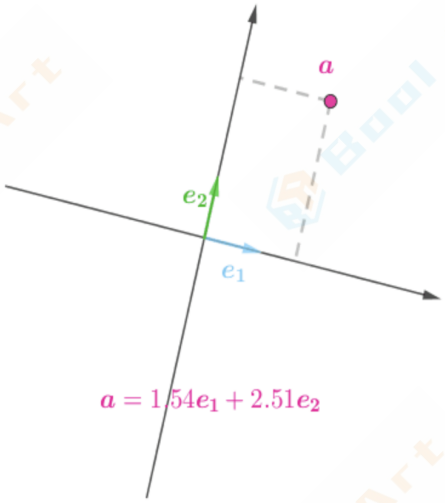
那么应该怎么降维呢？分析一下，从线性代数的角度来看，二维坐标系总有各自的标准正交基（也就是两两正交、模长为1的基），e_1, e_2：



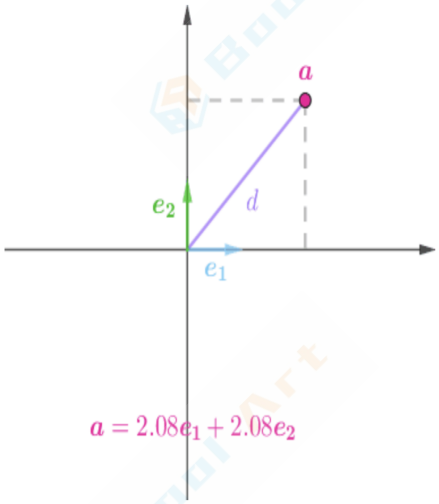
在某坐标系有一个点，$a = \begin{pmatrix} x \\ y \end{pmatrix}$，它表示在该坐标系下标准正交基$e_1, e_2$的线性组合：

$$\boldsymbol{a} = \begin{pmatrix} x \\ y \end{pmatrix} = x\boldsymbol{e_1} + y\boldsymbol{e_2}$$

只是在不同坐标系中，x，y的值会有所不同（旋转的坐标表示不同的坐标系）：



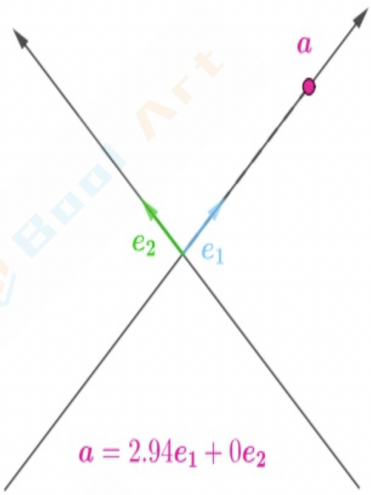
因为a到原点的距离d不会因为坐标系改变而改变：



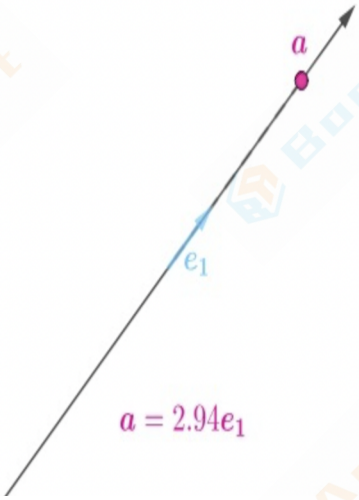
而：

$$d^2 = x^2 + y^2$$

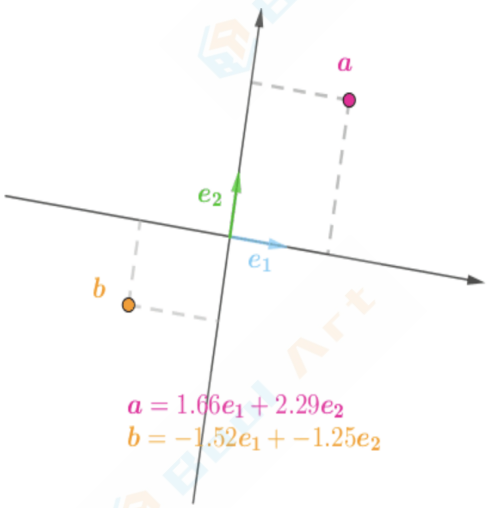
所以，在某坐标系下分配给x较多，那么分配给y的就必然较少，反之亦然。最极端的情况是，在某个坐标系下，全部分配给了x，使得y=0：



那么在这个坐标系中，就可以降维了，去掉e_2并不会丢失信息：



如果是两个点$a = \begin{pmatrix} x1 \\ y2 \end{pmatrix}$，$b = \begin{pmatrix} x2 \\ y2 \end{pmatrix}$，情况就复杂一些：



为了降维，应该选择尽量多分配给x_1, x_2，少分配给y_1, y_2的坐标系。

3 主元分析（PCA）

假设有如下数据：

	X	Y
a	a_1	b_1
b	a_2	b_2

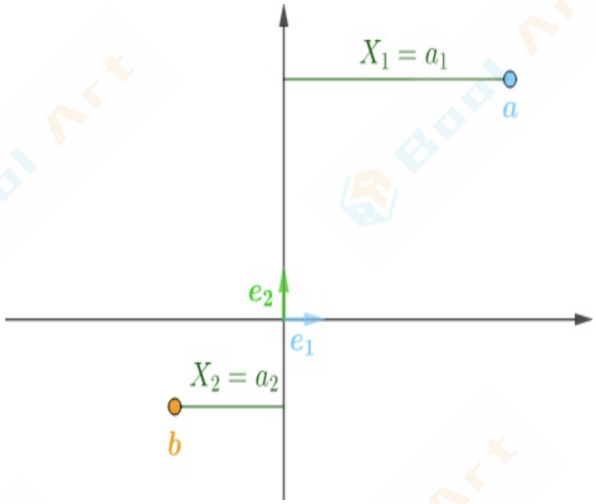
上面的数据这么解读，表示有两个点：

$$\boldsymbol{a} = \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix}$$

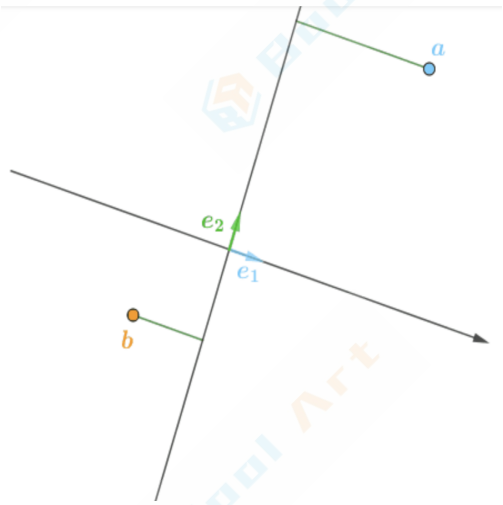
这两个点在初始坐标系下（也就是自然基$e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$，$e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$)下坐标值为：

$$\boldsymbol{a} = \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} a_2 \\ b_2 \end{pmatrix}$$

图示如下：



随着坐标系的不同，X_1, X_2的值会不断变化：



要想尽量多分配给X_1, X_2，借鉴最小二乘法的思想，就是让： $X_1^2 + X_2^2 = \sum_{i=0}^{2} X_i^2$最大。

要求这个问题，先看看X_1, X_2怎么表示，假设：

$$\boldsymbol{e_1} = \begin{pmatrix} e_{11} \\ e_{12} \end{pmatrix} \quad \boldsymbol{e_2} = \begin{pmatrix} e_{21} \\ e_{22} \end{pmatrix}$$

根据点积的几何意义（如何通俗地理解协方差和点积）有：

$$X_1 = \boldsymbol{a} \cdot \boldsymbol{e_1} = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdot \begin{pmatrix} e_{11} \\ e_{12} \end{pmatrix} = a_1 e_{11} + b_1 e_{12}$$

那么：

$$\begin{aligned} X_1^2 + X_2^2 &= (a_1 e_{11} + b_1 e_{12})^2 + (a_2 e_{11} + b_2 e_{12})^2 \\ &= a_1^2 e_{11}^2 + 2a_1 b_1 e_{11} e_{12} + b_1^2 e_{12}^2 + a_2^2 e_{11}^2 + 2a_2 b_2 e_{11} e_{12} + b_2^2 e_{12}^2 \\ &= (a_1^2 + a_2^2) e_{11}^2 + 2(a_1 b_1 + a_2 b_2) e_{11} e_{12} + (b_1^2 + b_2^2) e_{12}^2 \end{aligned}$$

上式其实是一个二次型（可以参看如何通俗地理解二次型）：

$$X_1^2+X_2^2=\boldsymbol{e}_1^{\mathrm{T}}\underbrace{\begin{pmatrix}a_1^2+a_2^2 & a_1b_1+a_2b_2\\ a_1b_1+a_2b_2 & b_1^2+b_2^2\end{pmatrix}}_{P}\boldsymbol{e}_1=\boldsymbol{e}_1^{\mathrm{T}}P\boldsymbol{e}_1$$

这里矩阵P就是二次型，是一个对称矩阵，可以进行如下的奇异值分解：

$$P=U\Sigma U^{\mathrm{T}}$$

U为正交矩阵，即$UU^T=I$。

而∑是对角矩阵：

$$\Sigma=\begin{pmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{pmatrix}$$

其中，σ_1,σ_2是奇异值，$\sigma_1>\sigma_2$。将P代回去：

$$\begin{aligned}X_1^2+X_2^2&=\boldsymbol{e}_1^{\mathrm{T}}P\boldsymbol{e}_1\\ &=\boldsymbol{e}_1^{\mathrm{T}}U\Sigma U^{\mathrm{T}}\boldsymbol{e}_1\\ &=(U^{\mathrm{T}}\boldsymbol{e}_1)^{\mathrm{T}}\Sigma(U^{\mathrm{T}}\boldsymbol{e}_1)\end{aligned}$$

因为U是正交矩阵，所以令：

$$\boldsymbol{n}=U^{\mathrm{T}}\boldsymbol{e}_1$$

所得的n也是单位向量，即：

$$\boldsymbol{n}=\begin{pmatrix}n_1\\ n_2\end{pmatrix}\implies n_1^2+n_2^2=1$$

继续回代：

$$\begin{aligned}X_1^2+X_2^2&=(U^{\mathrm{T}}\boldsymbol{e}_1)^{\mathrm{T}}\Sigma(U^{\mathrm{T}}\boldsymbol{e}_1)\\ &=\boldsymbol{n}^{\mathrm{T}}\Sigma\boldsymbol{n}\\ &=\begin{pmatrix}n_1 & n_2\end{pmatrix}\begin{pmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{pmatrix}\begin{pmatrix}n_1\\ n_2\end{pmatrix}\\ &=\sigma_1n_1^2+\sigma_2n_2^2\end{aligned}$$

最初求最大值的问题就转化为了：

$$X_1^2+X_2^2=\sum_{i=0}^{2}X_i^2 \text{ 最大} \iff \begin{cases}\sigma_1 n_1^2+\sigma_2 n_2^2 \text{ 最大}\\ n_1^2+n_2^2=1\\ \sigma_1>\sigma_2\end{cases}$$

因此可以推出要寻找的主元1，即：

$$\boldsymbol{n}=\begin{pmatrix}1\\0\end{pmatrix}=U^{\mathrm{T}}\boldsymbol{e_1} \implies \boldsymbol{e_1}=U\begin{pmatrix}1\\0\end{pmatrix}$$

总结下：

$$\boldsymbol{e_1}=\begin{cases}P=U\Sigma U^{\mathrm{T}}\\ \text{最大奇异值}\sigma_1\text{对应的奇异向量}\end{cases}$$

同样的思路可以求出：

$$\boldsymbol{e_2}=\begin{cases}P=U\Sigma U^{\mathrm{T}}\\ \text{最小奇异值}\sigma_2\text{对应的奇异向量}\end{cases}$$

4 协方差矩阵

上一小节的数据：

	X	Y
a	a_1	b_1
b	a_2	b_2

我们按行来解读，得到了两个向量a，b：

	X	Y
$\boldsymbol{a}:$ a	a_1	b_1
$\boldsymbol{b}:$ b	a_2	b_2

在这个基础上推出了矩阵：

$$P=\begin{pmatrix}a_1^2+a_2^2 & a_1b_1+a_2b_2\\ a_1b_1+a_2b_2 & b_1^2+b_2^2\end{pmatrix}$$

这个矩阵是求解主元1、主元2的关键。 如果我们按列来解读，可以得到两个向量X，Y：

	X	Y
a	a_1	b_1
b	a_2	b_2

即：

$$\boldsymbol{X}=\begin{pmatrix}a_1\\a_2\end{pmatrix}\quad\boldsymbol{Y}=\begin{pmatrix}b_1\\b_2\end{pmatrix}$$

那么刚才求出来的矩阵就可以表示为：

$$P=\begin{pmatrix}a_1^2+a_2^2 & a_1b_1+a_2b_2\\a_1b_1+a_2b_2 & b_1^2+b_2^2\end{pmatrix}=\begin{pmatrix}X\cdot X & X\cdot Y\\X\cdot Y & Y\cdot Y\end{pmatrix}$$

“中心化”后的样本方差：

$$Var(X)=\frac{1}{n}\sum_{i=1}^{n}X_i^2=\frac{1}{n}X\cdot X$$

样本协方差为：

$$Cov(X,Y)=\frac{1}{n}\sum_{i=1}^{n}X_iY_i=\frac{1}{n}X\cdot Y$$

两相比较可以得到一个新的矩阵，也就是协方差矩阵：

$$Q=\frac{1}{n}P=\begin{pmatrix}Var(X) & Cov(X,Y)\\Cov(X,Y) & Var(Y)\end{pmatrix}$$

P，Q都可以进行奇异值分解：

$$P=U\begin{pmatrix}\sigma_1 & 0\\0 & \sigma_2\end{pmatrix}U^{\mathrm{T}}\quad Q=\frac{1}{n}P=U\begin{pmatrix}\frac{\sigma_1}{n} & 0\\0 & \frac{\sigma_2}{n}\end{pmatrix}U^{\mathrm{T}}$$

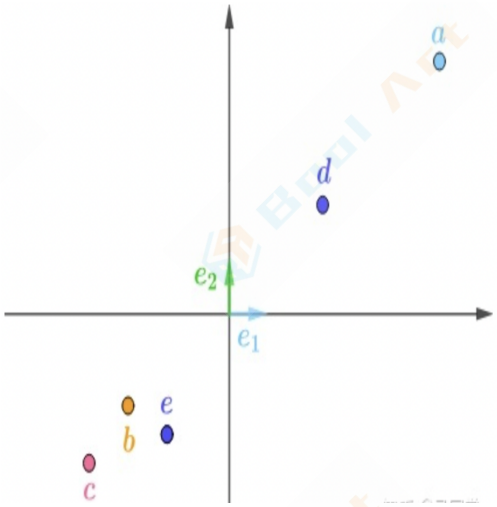
可见，协方差矩阵Q的奇异值分解和P相差无几，只是奇异值缩小了n倍，但是不妨碍奇异值之间的大小关系，所以在实际问题中，往往都是直接分解协方差矩阵Q。

5 实战

回到使用之前“中心化”了的数据：

	房价(百万元)	面积(百平米)
a	5.4	4.4
b	−2.6	−1.6
c	−3.6	−2.6
d	2.4	1.9
e	−1.6	−2.1

这些数据按行，在自然基下画出来就是：



按列解读得到两个向量：

$$\boldsymbol{X} = \begin{pmatrix} 5.4 \\ -2.6 \\ -3.6 \\ 2.4 \\ -1.6 \end{pmatrix} \quad \boldsymbol{Y} = \begin{pmatrix} 4.4 \\ -1.6 \\ -2.6 \\ 1.9 \\ -2.1 \end{pmatrix}$$

组成协方差矩阵：

$$Q = \begin{pmatrix} Var(X) & Cov(X,Y) \\ Cov(X,Y) & Var(Y) \end{pmatrix} = \frac{1}{5}\begin{pmatrix} X \cdot X & X \cdot Y \\ X \cdot Y & Y \cdot Y \end{pmatrix} = \frac{1}{5}\begin{pmatrix} 57.2 & 45.2 \\ 45.2 & 36.7 \end{pmatrix}$$

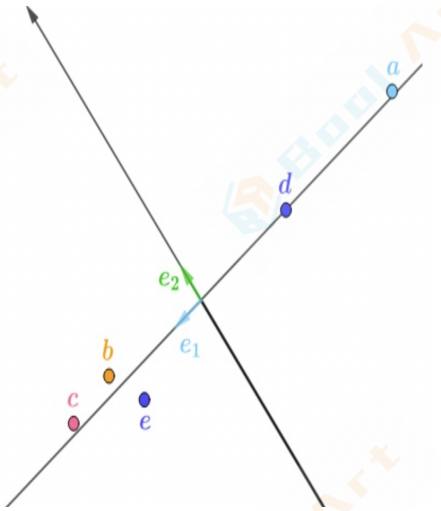
进行奇异值分解：

$$Q \approx \begin{pmatrix} -0.78 & -0.62 \\ -0.62 & 0.78 \end{pmatrix}\begin{pmatrix} 18.66 & 0 \\ 0 & 0.12 \end{pmatrix}\begin{pmatrix} -0.78 & -0.62 \\ -0.62 & 0.78 \end{pmatrix}$$

根据之前的分析，主元1应该匹配最大奇异值对应的奇异向量，主元2匹配最小奇异值对应的奇异向量，即：

$$\boldsymbol{e_1} = \begin{pmatrix} -0.78 \\ -0.62 \end{pmatrix} \quad \boldsymbol{e_2} = \begin{pmatrix} -0.62 \\ 0.78 \end{pmatrix}$$

以这两个为主元画出来的坐标系就是这样的：



如下算出新坐标，比如对于a：

$$X_1 = \boldsymbol{a} \cdot \boldsymbol{e_1} = -6.94 \quad X_2 = \boldsymbol{a} \cdot \boldsymbol{e_2} = 0.084$$

以此类推，得到新的数据表：

	主元1	主元2
a	-6.94	0.084
b	3.02	0.364
c	4.42	0.204
d	-3.05	-0.006
e	2.55	-0.646

主元2整体来看，数值很小，丢掉损失的信息也非常少，这样就实现了非理想情况下的降维。

二、PCA应用（wine data案例分析）

PCA（主成分分析）是一种无监督数据压缩技术，广泛应用于特征提取和降维。换言之，PCA技术就是在高维数据中寻找最大方差的方向，将这个方向投影到维度更小的新子空间。例如，将原数据向量x，通过构建d×k维变换矩阵W，映射到新的k维子空间，通常（k<<d）。 原数据d维向量空间$x = [x_1, x_2, \ldots, x_d], x \in R^d$ 经过 $xW, W \in R^{d \times k}$，得到新的k维向量空间$z = [z_1, z_2, \ldots, z_k], z \in R^k$。第一主成分有最大的方差，在PCA之前需要对特征进行标准化，保证所有特征在相同尺度下均衡。

方法步骤

- 标准化d维数据集
- 构建协方差矩阵。
- 将协方差矩阵分解为特征向量和特征值。
- 对特征值进行降序排列，相应的特征向量作为整体降序。
- 选择k个最大特征值的特征向量，。
- 根据提取的k个特征向量构造投影矩阵。
- d维数据经过变换获得k维。

1 Python代码实现

加载数据集

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
%matplotlib inline
sns.set_style('whitegrid')
```

```
Duplicate key in file PosixPath('/Users/yonna/opt/anaconda3/lib/python
3.8/site-packages/matplotlib/mpl-data/matplotlibrc'), line 253 ('font.
family:  sans-serif')
Duplicate key in file PosixPath('/Users/yonna/opt/anaconda3/lib/python
3.8/site-packages/matplotlib/mpl-data/matplotlibrc'), line 261 ('font.
sans-serif: DejaVu Sans, Bitstream Vera Sans, Computer Modern Sans Ser
if, Lucida Grande, Verdana, Geneva, Lucid, Arial, Helvetica, Avant Gar
de, sans-serif')
Duplicate key in file PosixPath('/Users/yonna/opt/anaconda3/lib/python
3.8/site-packages/matplotlib/mpl-data/matplotlibrc'), line 404 ('axes.
unicode_minus: True  # use Unicode for the minus symbol rather than hy
phen.  See')
```

In [2]:

```
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

In [3]:

```
from sklearn.datasets import load_wine
wine = load_wine()
```

In [4]:

```
col_names = list(wine.feature_names)
col_names.append('target')
df = pd.DataFrame(np.c_[wine.data, wine.target], columns=col_names)
df.head()
```

Out[4]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoi
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

加载sklearn自带的葡萄酒数据集，这些数据是对意大利同一地区种植的葡萄酒进行化学分析的结果，这些葡萄酒来自三个不同的品种。该分析确定了三种葡萄酒中每种葡萄酒中含有的十多种种成分的数量。不同种类的酒品，它的成分也有所不同，通过对这些成分的分析就可以对不同的特定的葡萄酒进行分类分析。数据集共有178个样

本、3种数据类别、每个样本的有13个属性。这13个属性分别是：酒精、苹果酸、灰、灰分的碱度、镁、总酚、黄酮类化合物、非黄烷类酚类、原花色素、颜色强度、色调、稀释葡萄酒的OD280 / OD315、脯氨酸。

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    float64
dtypes: float64(14)
memory usage: 19.6 KB
```

下面我们按照这一节开头提到的步骤自己用Python实现主成分分析。

1.1 提取主成分

In [6]:

```
#将数据按7:3划分为training-data和testing-data，并进行标准化处理。
x, y = df.iloc[:, 0:13].values, df.iloc[:, -1].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
                                                    stratify=None, random_state=258)
```

In [7]:

```
#标准化
sc = StandardScaler()
x_train_std = sc.fit_transform(x_train)
x_test_std = sc.fit_transform(x_test)
```

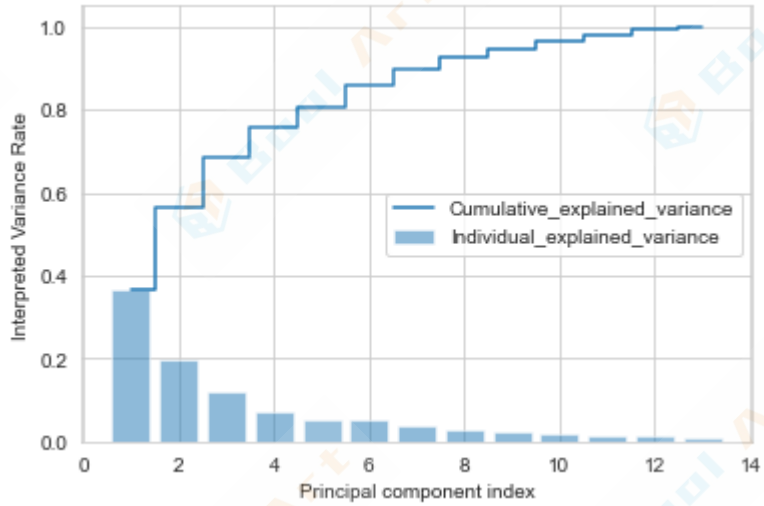
In [8]:

```
#构造协方差矩阵，得到特征向量和特征值
cov_matrix = np.cov(x_train_std.T)
eigen_val, eigen_vec = np.linalg.eig(cov_matrix)
#print("values\n ", eigen_val, "\nvector\n ", eigen_vec)# 可以打印看看
```

1.2 主成分方差可视化

In [9]:

```
# 解释方差比
tot = sum(eigen_val) # 总特征值和
var_exp = [(i / tot) for i in sorted(eigen_val, reverse=True)] # 计算解释方差比，降序
# print(var_exp)
cum_var_exp = np.cumsum(var_exp) # 累加方差比率
plt.bar(range(1, 14), var_exp, alpha=0.5,
        align='center', label='Individual_explained_variance') # 柱状 独立解释方差
plt.step(range(1, 14), cum_var_exp,
         where='mid', label='Cumulative_explained_variance') # 累加解释方差
plt.ylabel("Interpreted Variance Rate")
plt.xlabel("Principal component index")
plt.legend(loc='right')
plt.show()
```



可视化结果看出，第一二主成分占据大部分方差，接近60%。

1.3 特征变换

这一步需要构造之前讲到的投影矩阵，从高维d变换到低维空间k。 先将提取的特征对进行降序排列：

In [10]:

```
eigen_pairs = [(np.abs(eigen_val[i]), eigen_vec[:, i]) for i in range(len(eigen_val))
eigen_pairs.sort(key=lambda k: k[0], reverse=True) # (特征值，特征向量)降序排列
```

从上步骤可视化，选取第一二主成分作为最大特征向量进行构造投影矩阵。

In [11]:

```
w = np.hstack((eigen_pairs[0][1][:, np.newaxis], eigen_pairs[1][1][:, np.newaxis]))
# 降维投影矩阵w
```

In [12]:

```
#13×2维矩阵
print(w)
```

```
[[-0.11473272  0.47381421]
 [ 0.27689448  0.24842488]
 [ 0.01842648  0.31729288]
 [ 0.25232945 -0.0147614 ]
 [-0.09416619  0.29750141]
 [-0.38832218  0.10301302]
 [-0.41769499  0.02641197]
 [ 0.27817666  0.04376415]
 [-0.32536834  0.12202551]
 [ 0.12530589  0.49620259]
 [-0.31424164 -0.27540871]
 [-0.3784389  -0.12542706]
 [-0.26667111  0.39808504]]
```

In [13]:

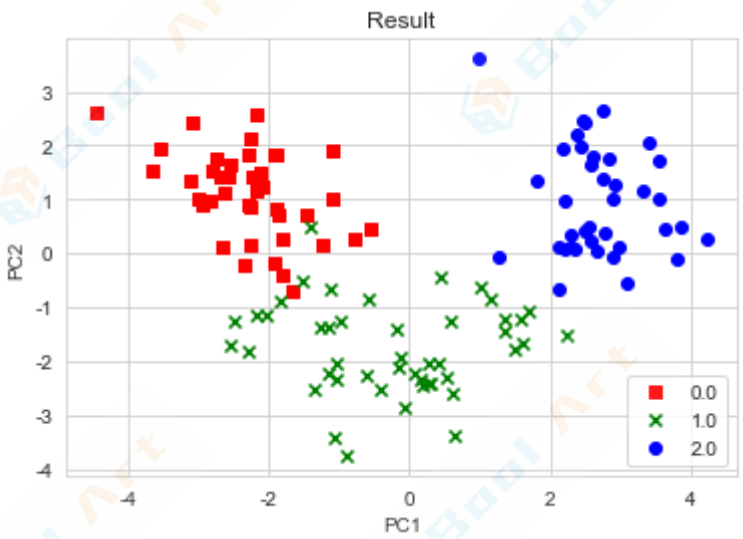
```
#这时，将原数据矩阵与投影矩阵相乘，转化为只有两个最大的特征主成分。
x_train_pca = x_train_std.dot(w)
```

1.4 数据分类结果

使用 matplotlib进行画图可视化

In [14]:

```
color = ['r', 'g', 'b']
marker = ['s', 'x', 'o']
for l, c, m in zip(np.unique(y_train), color, marker):
    plt.scatter(x_train_pca[y_train == l, 0],
                x_train_pca[y_train == l, 1],
                c=c, label=l, marker=m)
plt.title('Result')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(loc='lower right')
plt.show()
```



这时可以很直观区别3种不同类别！PCA是无监督学习技术，它的分类没有使用到样本标签，上面之所以看出3类不同标签，是画图时候自行添加的类别区分标签。

2 运用sklearn进行主成分分析(PCA)代码实现

sklearn中的PCA类相当于一个转换器，首先用训练数据来拟合模型，仍以wine data为例，通过逻辑回归转化样本数据，实现主成分分析以及特征提取，直接调用PCA类即可。

In [15]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
```

In [16]:

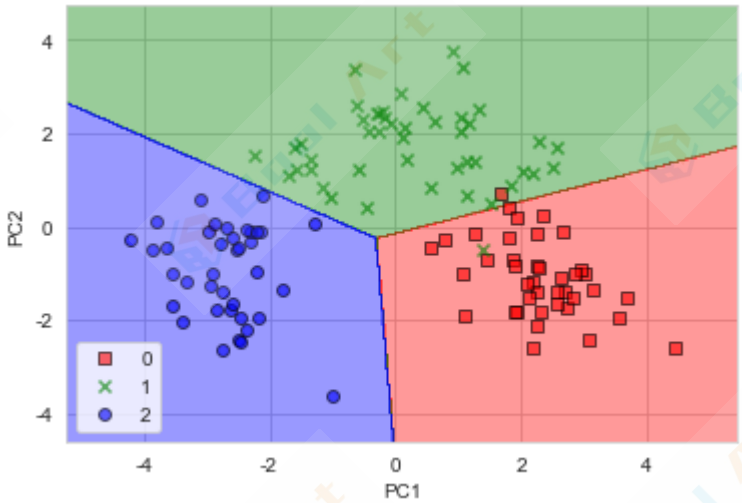
```
#为了在分类结果区别决策区域并可视化表示，这里编写plot_decision_region函数。
def plot_decision_regions(x, y, classifier, resolution=0.02):
    markers = ['s', 'x', 'o', '^', 'v']
    colors = ['r', 'g', 'b', 'gray', 'cyan']
    cmap = ListedColormap(colors[:len(np.unique(y))])

    x1_min, x1_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    x2_min, x2_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    z = z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, z, alpha=0.4, cmap=cmap)

    for idx, cc in enumerate(np.unique(y)):
        plt.scatter(x=x[y == cc, 0],
                    y=x[y == cc, 1],
                    alpha=0.6,
                    c=cmap(idx),
                    edgecolor='black',
                    marker=markers[idx],
                    label=cc)
```

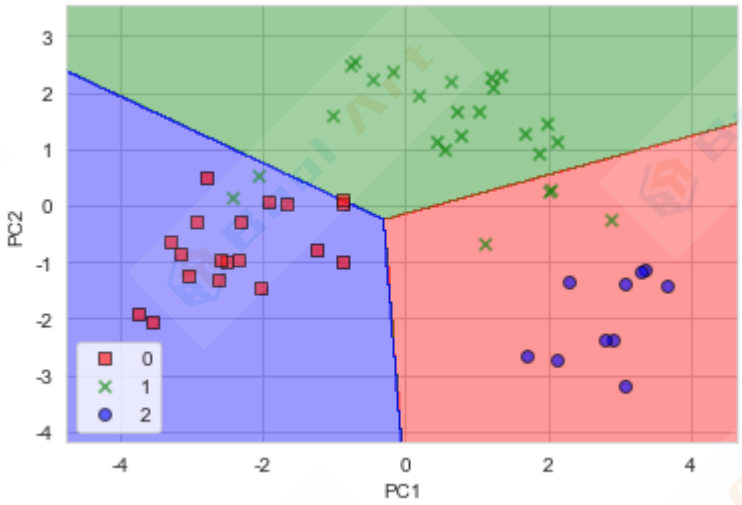
In [17]:

```
#10行代码完成数据集分类
#训练集
pca = PCA(n_components=2) # 前两个主成分
lr = LogisticRegression() # 逻辑回归来拟合模型
x_train_pca = pca.fit_transform(x_train_std)
x_test_pca = pca.fit_transform(x_test_std)
lr.fit(x_train_pca, y_train.astype('int'))
plot_decision_regions(x_train_pca, y_train.astype('int'), classifier=lr)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(loc='lower left')
plt.show()
```



In [18]:

```
#测试集
pca = PCA(n_components=2) # 前两个主成分
lr = LogisticRegression() # 逻辑回归来拟合模型
x_train_pca = pca.fit_transform(x_train_std)
x_test_pca = pca.fit_transform(x_test_std)*(-1)#预测时候特征向量正负问题，乘-1反转镜像
lr.fit(x_train_pca, y_train.astype('int'))
plot_decision_regions(x_test_pca, y_test.astype('int'), classifier=lr)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(loc='lower left')
plt.show()
```



使用逻辑回归对训练数据进行拟合，建立模型。训练集上的分类效果还是很不错，跟上面自己实现的PCA几乎一样。

在上面的例子中，PCA分析不是简单地选取了2个变化最大的变量，而是先把原始的13个变量做一个评估，计算各个变量各自的方差和两两变量的协方差，得到一个协方差矩阵。

在这个协方差矩阵中，对角线的值为每一个变量的方差，其它值为每两个变量的协方差。随后对原变量的协方差矩阵对角化处理，即求解其特征值和特征向量。

原变量与特征向量的乘积即为新变量；新变量的协方差矩阵为对角协方差矩阵且对角线上的方差由大到小排列；然后从新变量中选择信息最丰富也就是方差最大的的前2个新变量也就是主成分，用以可视化。

我们可以发现，通过主成分分析，将13维的问题降到2维。只选取两个主成分进行分析，仍然可以达到最终的分类效果。PCA可以很好地达到去除特征冗余的目的，实现降维和去燥。

基于sklearn的主成分分析代码实现，使用PCA进行无监督数据降维，编码效率提高了许多。

3 总结

在这个实例中，我们学习了用python实现PCA的步骤和过程，了解到了：

- 主成分分析原理以及如何使用它来减少问题的复杂性。
- 基于sklearn的主成分分析代码实现，使用PCA进行无监督数据降维。

参考资料:

- scikit-learn Doc (http://scikit-learn.org/stable/modules/decomposition.html#pca)
- scikit-learn Parameters (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA)
- How to Calculate Principal Component Analysis (PCA) from Scratch in Python (https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/)