15.2 主成分分析PCA

主成分分析(Principal Component Analysis，PCA)是在降维中使用特别广泛的算法。在使用主成分 分析降维的时候没有束缚，不像线性判别分析，必须要有数据标签，只要拿到数据，没有标签也可以用 主成分分析进行降维。所以应该先有一个直观的认识，主成分分析本质上属于**无监督算法**，这也是它流 行的主要原因。

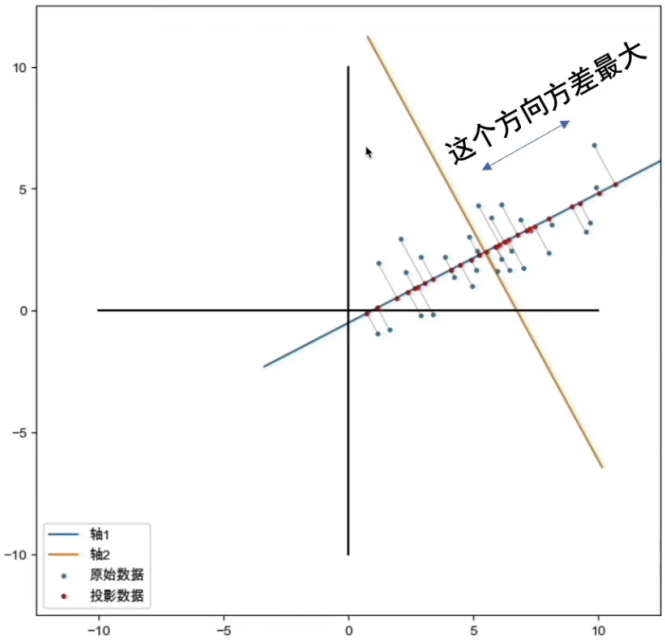
B站视频：https://www.bilibili.com/video/BV1E5411E71z (https://www.bilibili.com/video/BV1E5411E71z)

15.2.1 PCA降维基本知识点

考虑将一组二维数据进行降维，如何选取这个一维坐标系使得能保留的信息最多呢？如果降维后都到同一个点，那就没有保留任何信息，而要保留更多的信息，就得尽可能有区分度（方差尽可能大）

目标：只保留一个轴的时候（二维降到一维），信息保留最多

怎么样最好：
找到数据分布最分散的方向（方差最大），作为主成分（坐标轴）



找坐标系

为了让大家更好地理解主成分分析，下面介绍一些基本概念。

用基表示向量

(1)向量的表示。假设有向量(3，2)，如图15-7所示。为什么向量可以表示为(3，2)，这是在 直角坐标系中的表示，如果坐标系变了，向量的表示形式也要发生变换。 实际上该向量可以表示成线性组合3·(1，0)T+2·(0，1)T，其中(1，0)和(0，1)就称为二维空间中 的一组基。

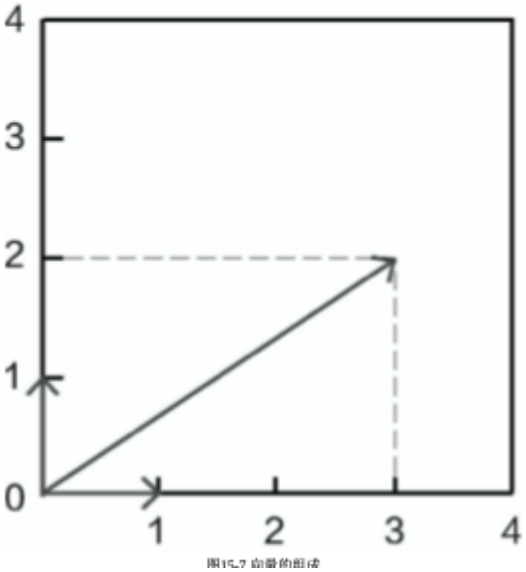


图15-7 向量的组成

基的变化

(2)基变换。大家常见的坐标系都是正交的，即内积为0，两两相互垂直，并且线性无关。为什么 基都是这样的呢?如果不垂直，那么肯定线性相关，能用一个表示另一个，此时基就会失去意义，所以 基的出发点就是要正交。

基也可以进行变换，将一个向量从一组基变换到另一组基中。例如新的坐标系的两个基分别是$(1/\sqrt{2},\ 1/\sqrt{2})$，$(-1/\sqrt{2},\ 1/\sqrt{2})$，因此向量（3，2）映射到这个新的坐标系中，可以通过下面变换实现：

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}\begin{pmatrix} 3 \\ 2 \end{pmatrix}=\begin{pmatrix} 5/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

方差和协方差

(3)方差和协方差。方差(variance)相当于特征辨识度，其值越大越好。协方差(covariance)就 是不同特征之间的相关程度，协方差的计算式为:

$$\mathrm{cov}(a,b)=\frac{1}{m-1}\sum_{i=1}^{m}(a_i-\mu)(b_i-v)$$

如果两个变量的变化趋势相同，例如随着身高的增长，体重也增长，此时它们的协方差值就会比较 大，表示正相关。而方差又描述了各自的辨识能力，接下来就要把这些知识点穿插在一起。

15.2.2 PCA优化目标求解

对于降维任务，无非就是将原始数据特征投影到一个更合适的空间，结合基的概念，这就相当于由 一组基变换到另一组基，变换的过程要求特征变得更有价值，也就是方差能够更大。所以现在已经明确 基本目标了:找到一组基，使得变换后的特征方差越大越好。

假设找到了第一个合适的投影方向，这个方向能够使得方差最大，对于降维任务来说，一般情况下 并不是降到一维，接下来肯定要找方差第二大的方向。方差第二大的方向理论上应该与第一方向非常接 近，甚至重合，这样才能保证方差最大，如图15-8所示。

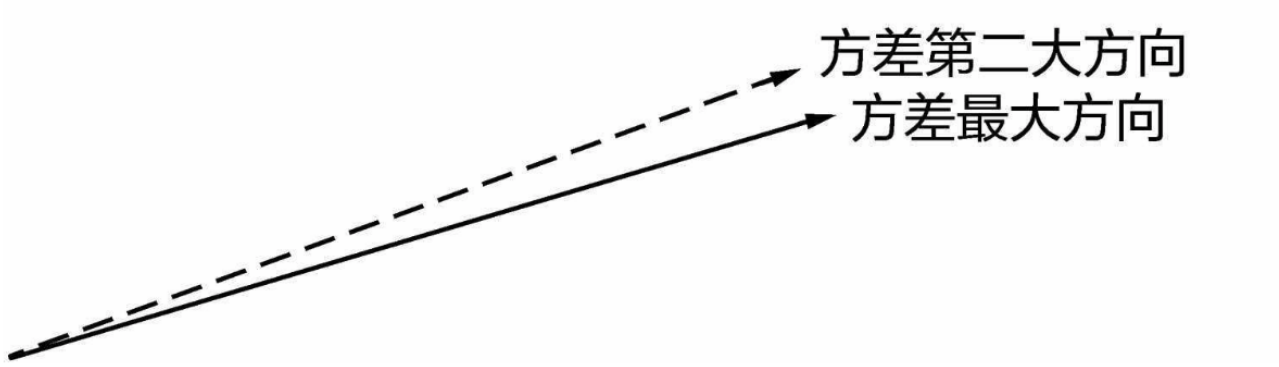


图15-8 方差方向选择

在这种情况下，看似可以得到无数多个方差非常大的方向，但是想一想它们能组成想要的基吗?不 能，因为没有满足基的最基本要求——线性无关，也就是相互垂直正交。所以在寻找方差最大的方向的 同时，还要使得各个投影方向能够正交，即协方差应当等于0，表示完全独立无关。所以在选择基的时 候，一方面要尽可能地找方差的最大方向，另一方面要在其正交方向上继续寻找方差第二大的方向，以此类推。

解释PCA中要求解的目标后，接下来就是在数学上将它表达出来。先来看一下协方差矩阵，为了简便，可以把数据中各个特征的均值默认为0，也可以认为数据已经进行过标准化处理。其计算式如下:

$$X=\begin{pmatrix} a_1 & a_2 & \dots & a_m \\ b_1 & b_2 & \dots & b_m \end{pmatrix}$$

其中，X为实际的数据。包含2个特征a和b，一共有m个样本。

此时协方差矩阵为:

$$\frac{1}{m}XX^{\mathrm{T}}=\begin{pmatrix} \frac{1}{m}\sum_{i=1}^{m}a_i^2 & \frac{1}{m}\sum_{i=1}^{m}a_ib_i \\ \frac{1}{m}\sum_{i=1}^{m}a_ib_i & \frac{1}{m}\sum_{i=1}^{m}b_i \end{pmatrix}$$

先观察一下协方差矩阵结果，其主对角线上的元素就是各个特征的方差(均值为0时)，而非主对角 线的上元素恰好是特征之间的协方差。按照目标函数的要求，首先应当使得方差越大越好，并且确保协 方差为0，这就需要对协方差矩阵做对角化。

从一个n行n列的实对称矩阵中一定可以找到n个单位正交特征向量E=(e1,e2 ,... ,en)，以完成对角化的 操作:

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

式(15.21)中的协方差矩阵恰好满足上述要求。假设需要将一组N维向量降为K维(K大于0，小于 N)，目标是选择K个单位正交基，使原始数据变换到这组基上后，各字段两两间协方差为0，各字段本身 的方差尽可能大。当得到其协方差矩阵后，对其进行对角化操作，即可使得除主对角线上元素之外都为 0。

其中对角线上的元素就是矩阵的特征值，这与线性判别分析很像，还是先把特征值按从大到小的顺 序进行排列，找到前K个最大特征值对应的特征向量，接下来就是进行投影变换。

按照给定PCA优化目标，基本流程如下。

第1步:数据预处理，只有数值数据才可以进行PCA降维。

第2步:计算样本数据的协方差矩阵。

第3步:求解协方差矩阵的特征值和特征向量。

第4步:将特征值按照从大到小的顺序排列，选择其中较大的K个，然后将其对应的K个特征向量组成投影矩阵。

第5步:将样本点投影计算，完成PCA降维任务。

15.2.3 Python实现PCA降维

接下来通过一个实例介绍如何使用PCA处理实际问题，同样使用鸢尾花数据集，目的依旧是完成降 维任务，下面就来看一下PCA是怎么实现的。

第1步:导入数据。

In [2]:

```
import numpy as np
import pandas as pd

# 读取数据集
df = pd.read_csv('iris.data')
# 原始数据没有给定列名的时候需要我们自己加上
df.columns=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class']
df.head()
```

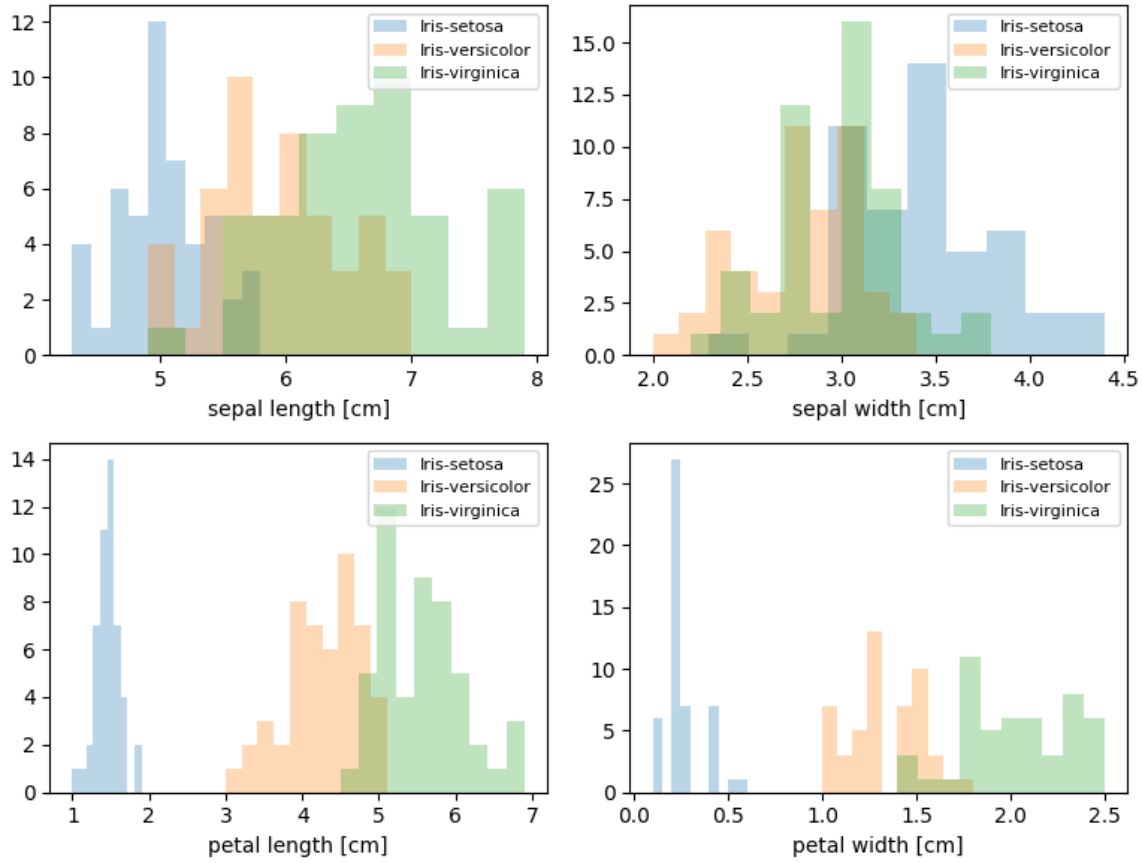
Out[2]:

	sepal_len	sepal_wid	petal_len	petal_wid	class
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

第2步:展示数据特征。

In [3]:

```
# 把数据分成特征和标签

X = df.iloc[:,0:4].values
y = df.iloc[:,4].values

from matplotlib import pyplot as plt

# 展示我们标签用的
label_dict = {1: 'Iris-Setosa',
              2: 'Iris-Versicolor',
              3: 'Iris-Virgnica'}

# 展示特征用的
feature_dict = {0: 'sepal length [cm]',
                1: 'sepal width [cm]',
                2: 'petal length [cm]',
                3: 'petal width [cm]'}

# 指定绘图区域大小
plt.figure(figsize=(8, 6))
for cnt in range(4):
    # 这里用子图来呈现4个特征
    plt.subplot(2, 2, cnt+1)
    for lab in ('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'):
        plt.hist(X[y==lab, cnt],
                 label=lab,
                 bins=10,
                 alpha=0.3,)
    plt.xlabel(feature_dict[cnt])
    plt.legend(loc='upper right', fancybox=True, fontsize=8)

plt.tight_layout()
plt.show()
```



上述代码可以生成图15-9的输出。可以看出，有些特征区别能力较强，能把3种花各自呈现出来;有 些特征区别能力较弱，部分特征数据样本混杂在一起。

第3步:数据的标准化。一般情况下，在进行训练前，数据经常需要进行标准化处理，可以使用sklearn库中的StandardScaler方法进行标准化处理，代码如下:

In [4]:

```
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)
```

计算协方差矩阵

In [5]:

```
mean_vec = np.mean(X_std, axis=0)
cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
print('协方差矩阵 \n%s' %cov_mat)
```

协方差矩阵

```
[[ 1.00675676 -0.10448539  0.87716999  0.82249094]
 [-0.10448539  1.00675676 -0.41802325 -0.35310295]
 [ 0.87716999 -0.41802325  1.00675676  0.96881642]
 [ 0.82249094 -0.35310295  0.96881642  1.00675676]]
```

或者可以直接使用Numpy工具包来计算协方差，结果是一样的:

In [6]:

```
print('直接用 NumPy 计算协方差矩阵: \n%s' %np.cov(X_std.T))
```

```
直接用 NumPy 计算协方差矩阵:
[[ 1.00675676 -0.10448539  0.87716999  0.82249094]
 [-0.10448539  1.00675676 -0.41802325 -0.35310295]
 [ 0.87716999 -0.41802325  1.00675676  0.96881642]
 [ 0.82249094 -0.35310295  0.96881642  1.00675676]]
```

第5步:求特征值与特征向量。

In [7]:

```
cov_mat = np.cov(X_std.T)

eig_vals, eig_vecs = np.linalg.eig(cov_mat)

print('特征向量 \n%s' %eig_vecs)
print('\n特征值 \n%s' %eig_vals)
```

```
特征向量
[[ 0.52308496 -0.36956962 -0.72154279  0.26301409]
 [-0.25956935 -0.92681168  0.2411952  -0.12437342]
 [ 0.58184289 -0.01912775  0.13962963 -0.80099722]
 [ 0.56609604 -0.06381646  0.63380158  0.52321917]]

特征值
[2.92442837 0.93215233 0.14946373 0.02098259]
```

第6步:按照特征值大小进行排序。

In [8]:

```
# 把特征值和特征向量对应起来
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]
print (eig_pairs)
print ('----------')
# 把它们按照特征值大小进行排序
eig_pairs.sort(key=lambda x: x[0], reverse=True)

# 打印排序结果
print('特征值又大到小排序结果:')
for i in eig_pairs:
    print(i[0])
```

```
[(2.9244283691111117, array([ 0.52308496, -0.25956935,  0.58184289,
0.56609604])), (0.9321523302535063, array([-0.36956962, -0.92681168,
-0.01912775, -0.06381646])), (0.14946373489813355, array([-0.7215427
9,  0.2411952 ,  0.13962963,  0.63380158])), (0.020982592764270655,
array([ 0.26301409, -0.12437342, -0.80099722,  0.52321917]))]
----------
特征值又大到小排序结果:
2.9244283691111117
0.9321523302535063
0.14946373489813355
0.020982592764270655
```

第7步:计算累加贡献率。同样可以用累加的方法，将特征向量累加起来，当其超过一定百分比 时，就选择其为降维后的维度大小:

In [9]:

```
# 计算累加结果
tot = sum(eig_vals)
var_exp = [(i / tot)*100 for i in sorted(eig_vals, reverse=True)]
print (var_exp)
cum_var_exp = np.cumsum(var_exp)
cum_var_exp
```

```
[72.62003332692032, 23.147406858644146, 3.7115155645845292, 0.521044
2498510169]
```

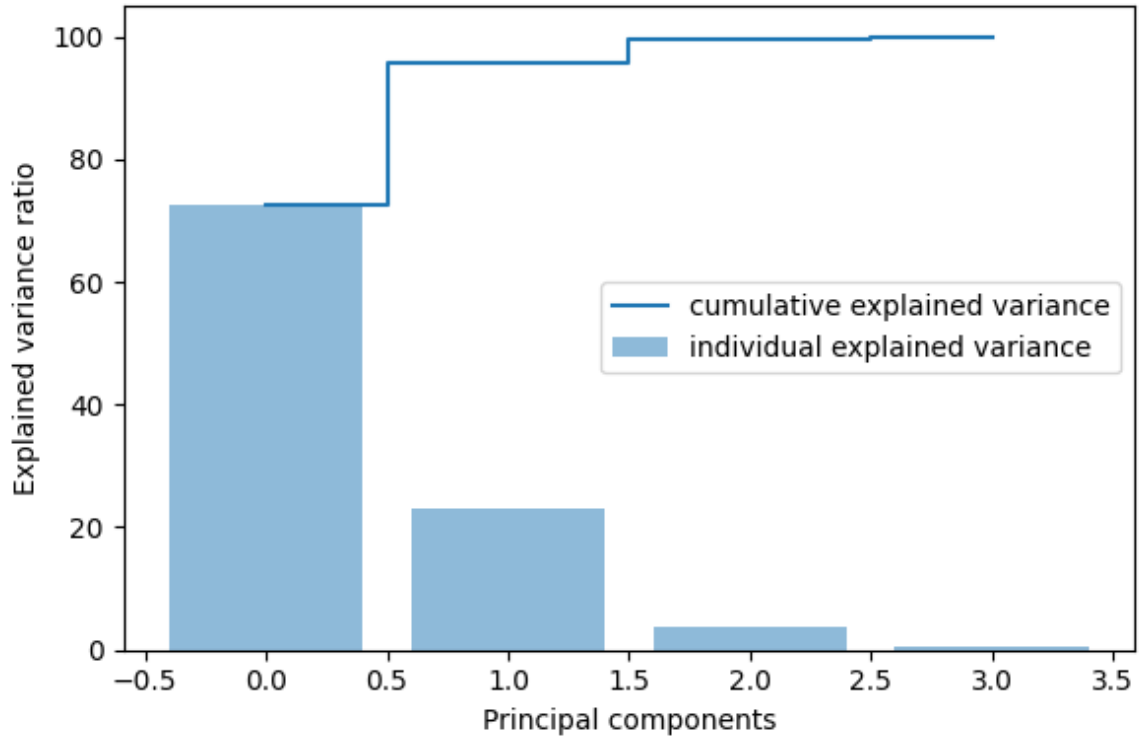
Out[9]:

```
array([ 72.62003333,  95.76744019,  99.47895575, 100.        ])
```

可以发现，使用前两个特征值时，其对应的累计贡献率已经超过95%，所以选择降到二维。也可以通 过画图的形式，这样更直接:

In [12]:

```
plt.figure(figsize=(6, 4))

plt.bar(range(4), var_exp, alpha=0.5, align='center',
            label='individual explained variance')
plt.step(range(4), cum_var_exp, where='mid',
             label='cumulative explained variance')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



第8步:完成PCA降维。接下来把特征向量组合起来完成降维工作:

In [13]:

```
matrix_w = np.hstack((eig_pairs[0][1].reshape(4,1),
                      eig_pairs[1][1].reshape(4,1)))

print('Matrix W:\n', matrix_w)
```

```
Matrix W:
 [[ 0.52308496 -0.36956962]
 [-0.25956935 -0.92681168]
 [ 0.58184289 -0.01912775]
 [ 0.56609604 -0.06381646]]
```

In [14]:

```
Y = X_std.dot(matrix_w)
Y
```

Out[14]:

```
array([[-2.10795032,  0.64427554],
       [-2.38797131,  0.30583307],
       [-2.32487909,  0.56292316],
       [-2.40508635, -0.687591  ],
       [-2.08320351, -1.53025171],
       [-2.4636848 , -0.08795413],
       [-2.25174963, -0.25964365],
       [-2.3645813 ,  1.08255676],
       [-2.20946338,  0.43707676],
       [-2.17862017, -1.08221046],
       [-2.34525657, -0.17122946],
       [-2.24590315,  0.6974389 ],
       [-2.66214582,  0.92447316],
       [-2.2050227 , -1.90150522],
       [-2.25993023, -2.73492274],
       [-2.21591283, -1.52588897],
       [-2.20705382, -0.52623535],
       [-1.9077081 , -1.4415791 ],
       [-2.35411558, -1.17088308],
       [-1.93202643, -0.44083479],
       [-2.21942518, -0.96477499],
       [-2.79116421, -0.50421849],
       [-1.83814105, -0.11729122],
       [-2.24572458, -0.17450151],
       [-1.97825353,  0.59734172],
       [-2.06935091, -0.27755619],
       [-2.18514506, -0.56366755],
       [-2.15824269, -0.34805785],
       [-2.28843932,  0.30256102],
       [-2.16501749,  0.47232759],
       [-1.8491597 , -0.45547527],
       [-2.62023392, -1.84237072],
       [-2.44885384, -2.1984673 ],
       [-2.20946338,  0.43707676],
       [-2.23112223,  0.17266644],
       [-2.06147331, -0.6957435 ],
       [-2.20946338,  0.43707676],
       [-2.45783833,  0.86912843],
       [-2.1884075 , -0.30439609],
       [-2.30357329, -0.48039222],
       [-1.89932763,  2.31759817],
       [-2.57799771,  0.4400904 ],
       [-1.98020921, -0.50889705],
       [-2.14679556, -1.18365675],
       [-2.09668176,  0.68061705],
       [-2.39554894, -1.16356284],
       [-2.41813611,  0.34949483],
       [-2.24196231, -1.03745802],
       [-2.22484727, -0.04403395],
       [ 1.09225538, -0.86148748],
       [ 0.72045861, -0.59920238],
       [ 1.2299583 , -0.61280832],
       [ 0.37598859,  1.756516  ],
       [ 1.05729685,  0.21303055],
       [ 0.36816104,  0.58896262],
       [ 0.73800214, -0.77956125],
       [-0.52021731,  1.84337921],
       [ 0.9113379 , -0.02941906],
       [-0.01292322,  1.02537703],
```

```
       [-0.15020174,  2.65452146],
       [ 0.42437533,  0.05686991],
       [ 0.52894687,  1.77250558],
       [ 0.70241525,  0.18484154],
       [-0.05385675,  0.42901221],
       [ 0.86277668, -0.50943908],
       [ 0.33388091,  0.18785518],
       [ 0.13504146,  0.7883247 ],
       [ 1.19457128,  1.63549265],
       [ 0.13677262,  1.30063807],
       [ 0.72711201, -0.40394501],
       [ 0.45564294,  0.41540628],
       [ 1.21038365,  0.94282042],
       [ 0.61327355,  0.4161824 ],
       [ 0.68512164,  0.06335788],
       [ 0.85951424, -0.25016762],
       [ 1.23906722,  0.08500278],
       [ 1.34575245, -0.32669695],
       [ 0.64732915,  0.22336443],
       [-0.06728496,  1.05414028],
       [ 0.10033285,  1.56100021],
       [-0.00745518,  1.57050182],
       [ 0.2179082 ,  0.77368423],
       [ 1.04116321,  0.63744742],
       [ 0.20719664,  0.27736006],
       [ 0.42154138, -0.85764157],
       [ 1.03691937, -0.52112206],
       [ 1.015435  ,  1.39413373],
       [ 0.0519502 ,  0.20903977],
       [ 0.25582921,  1.32747797],
       [ 0.25384813,  1.11700714],
       [ 0.60915822, -0.02858679],
       [ 0.31116522,  0.98711256],
       [-0.39679548,  2.01314578],
       [ 0.26536661,  0.85150613],
       [ 0.07385897,  0.17160757],
       [ 0.20854936,  0.37771566],
       [ 0.55843737,  0.15286277],
       [-0.47853403,  1.53421644],
       [ 0.23545172,  0.59332536],
       [ 1.8408037 , -0.86943848],
       [ 1.13831104,  0.70171953],
       [ 2.19615974, -0.54916658],
       [ 1.42613827,  0.05187679],
       [ 1.8575403 , -0.28797217],
       [ 2.74511173, -0.78056359],
       [ 0.34010583,  1.5568955 ],
       [ 2.29180093, -0.40328242],
       [ 1.98618025,  0.72876171],
       [ 2.26382116, -1.91685818],
       [ 1.35591821, -0.69255356],
       [ 1.58471851,  0.43102351],
       [ 1.87342402, -0.41054652],
       [ 1.23656166,  1.16818977],
       [ 1.45128483,  0.4451459 ],
       [ 1.58276283, -0.67521526],
       [ 1.45956552, -0.25105642],
       [ 2.43560434, -2.55096977],
       [ 3.29752602,  0.01266612],
```

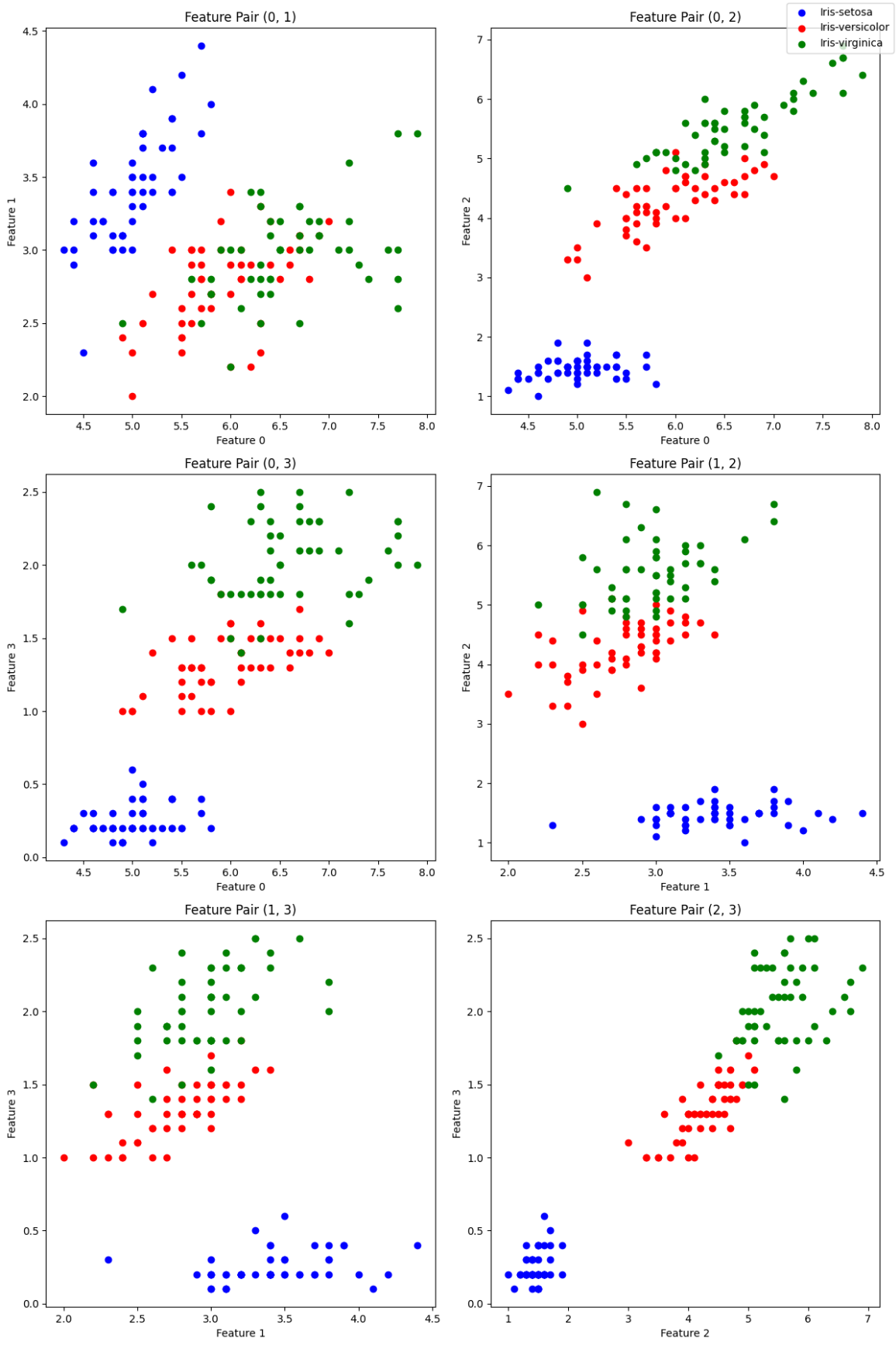
```
       [ 1.23377366,  1.71954411],
       [ 2.03218282, -0.90334021],
       [ 0.95980311,  0.57047585],
       [ 2.88717988, -0.38895776],
       [ 1.31405636,  0.48854962],
       [ 1.69619746, -1.01153249],
       [ 1.94868773, -0.99881497],
       [ 1.1574572 ,  0.31987373],
       [ 1.007133  , -0.06550254],
       [ 1.7733922 ,  0.19641059],
       [ 1.85327106, -0.55077372],
       [ 2.4234788 , -0.2397454 ],
       [ 2.31353522, -2.62038074],
       [ 1.84800289,  0.18799967],
       [ 1.09649923,  0.29708201],
       [ 1.1812503 ,  0.81858241],
       [ 2.79178861, -0.83668445],
       [ 1.57340399, -1.07118383],
       [ 1.33614369, -0.420823  ],
       [ 0.91061354, -0.01965942],
       [ 1.84350913, -0.66872729],
       [ 2.00701161, -0.60663655],
       [ 1.89319854, -0.68227708],
       [ 1.13831104,  0.70171953],
       [ 2.03519535, -0.86076914],
       [ 1.99464025, -1.04517619],
       [ 1.85977129, -0.37934387],
       [ 1.54200377,  0.90808604],
       [ 1.50925493, -0.26460621],
       [ 1.3690965 , -1.01583909],
       [ 0.94680339,  0.02182097]])
```

输出结果显示，使用PCA降维算法把原数据矩阵从150×4降到150×2。

第9步:可视化对比降维前后数据的分布。由于数据具有4个特征，无法在平面图中显示，因此只使 用两维特征显示数据，代码如下:

In [25]:

```
# 假设我们已加载鸢尾花数据集和处理好的特征矩阵 X 和标签 y
# 这里 X 是形状为 (150, 4) 的特征矩阵，y 是形状为 (150,) 的标签
# label_dict 是类别标签的字典，例如：{'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
# 反向的字典可以是 {0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'}

def plot_feature_combinations(X, y, label_dict):
    feature_pairs = [(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)]  # 特征组合
    fig, axes = plt.subplots(3, 2, figsize=(12, 18))  # 创建 3x2 的子图
    axes = axes.ravel()  # 将子图数组展平，方便索引

    for i, (x_idx, y_idx) in enumerate(feature_pairs):
        ax = axes[i]
        for lab, col in zip(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'),
                            ('blue', 'red', 'green')):
            ax.scatter(X[y == lab, x_idx],
                       X[y == lab, y_idx],
                       label=lab,
                       c=col)

        # 设置 x 和 y 轴标签
        ax.set_xlabel(f'Feature {x_idx}')
        ax.set_ylabel(f'Feature {y_idx}')
        ax.set_title(f'Feature Pair ({x_idx}, {y_idx})')

    # 添加图例到最后一个子图中
    handles, labels = ax.get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper right', fancybox=True)

    plt.tight_layout()
    plt.show()

# 调用函数来绘制图像
# 假设 X 是你的特征矩阵，y 是类别标签
plot_feature_combinations(X, y, label_dict)
```

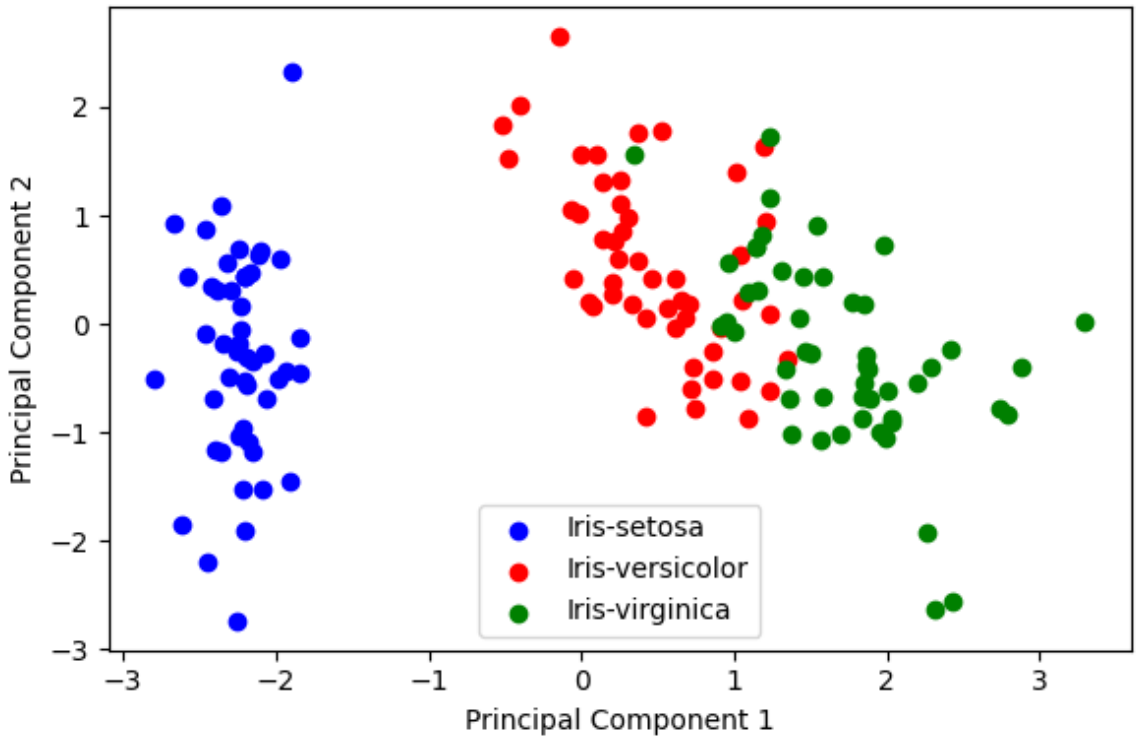


上面代码只使用前两个特征显示3类数据，如图15-11所示，看起来3类鸢尾花相互交叠在一起，不容 易区分开。

下面看看使用PCA降维后的情况，代码如下:

In [19]:

```
plt.figure(figsize=(6, 4))
for lab, col in zip(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'),
                    ('blue', 'red', 'green')):
    plt.scatter(Y[y==lab, 0],
                Y[y==lab, 1],
                label=lab,
                c=col)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='lower center')
plt.tight_layout()
plt.show()
```



本章总结

本章介绍了两种非常实用的降维方法:线性判别分析和主成分分析。其中线性判别分析是有监督算 法，需要有标签才能计算;而主成分析是无监督算法，无须标签，直接就可以对数据进行分析。那么， 在实际建模任务中，到底用哪种降维方法呢?没有固定的模式，需要大家通过实验对比确定，例如，取 得一份数据后可以分多步走，以对比不同策略的结果。

降维可以大大减少算法的计算量，加快程序执行的速度，遇到特征非常多的数据时就可以大显身 手。但是降维算法本身最大的问题就是降维后得到结果的物理含义很难解释，只能说这就是计算机看来 最好的降维特征，而不能具体化其含义。此时如果想进一步对结果进行分析就有些麻烦，因为其中每一 个特征指标的含义都只是数值，而没有具体指代。

Python案例中使用非常简单的鸢尾花数据集，按照原理推导的流程一步步完成了整个任务，大家在练 习的时候也可以选用稍微复杂一点的数据集来复现算法。

测试降到1维

In [16]:

```
# 测试降到1维
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)

print('特征向量 \n%s' %eig_vecs)
print('\n特征值 \n%s' %eig_vals)

# 把特征值和特征向量对应起来
# 测试降到1维
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]
matrix_w1= np.hstack((eig_pairs[0][1].reshape(4,1)))

print('Matrix W:\n', matrix_w1)
```

特征向量

```
[[ 0.52308496 -0.36956962 -0.72154279  0.26301409]
 [-0.25956935 -0.92681168  0.2411952  -0.12437342]
 [ 0.58184289 -0.01912775  0.13962963 -0.80099722]
 [ 0.56609604 -0.06381646  0.63380158  0.52321917]]
```

特征值

```
[2.92442837 0.93215233 0.14946373 0.02098259]
Matrix W:
 [ 0.52308496 -0.25956935  0.58184289  0.56609604]
```

In [17]:

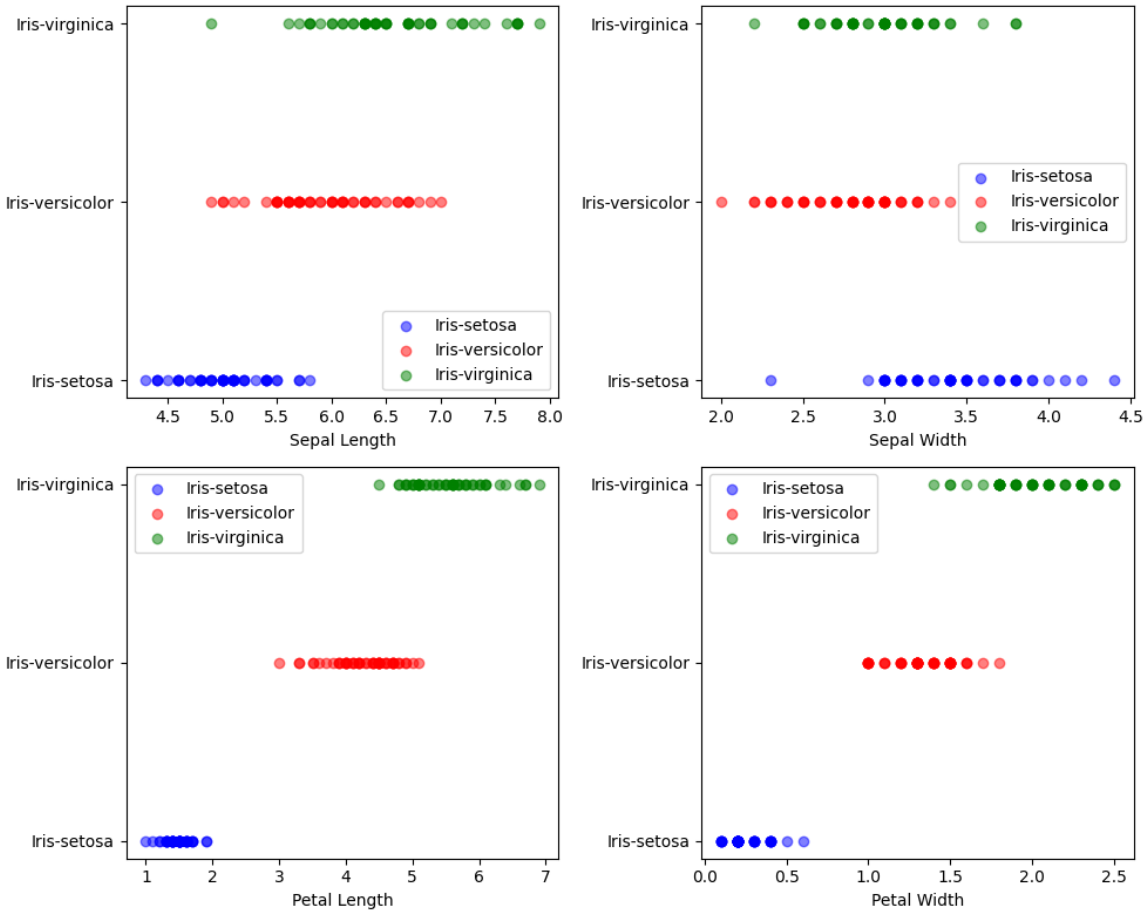
```
Y1 = X_std.dot(matrix_w1)
Y1
```

Out[17]:

```
array([-2.10795032, -2.38797131, -2.32487909, -2.40508635, -2.083203
51,
       -2.4636848 , -2.25174963, -2.3645813 , -2.20946338, -2.178620
17,
       -2.34525657, -2.24590315, -2.66214582, -2.2050227 , -2.259930
23,
       -2.21591283, -2.20705382, -1.9077081 , -2.35411558, -1.932026
43,
       -2.21942518, -2.79116421, -1.83814105, -2.24572458, -1.978253
53,
       -2.06935091, -2.18514506, -2.15824269, -2.28843932, -2.165017
49,
       -1.8491597 , -2.62023392, -2.44885384, -2.20946338, -2.231122
23,
       -2.06147331, -2.20946338, -2.45783833, -2.1884075 , -2.303573
29,
       -1.89932763, -2.57799771, -1.98020921, -2.14679556, -2.096681
76,
       -2.39554894, -2.41813611, -2.24196231, -2.22484727,  1.092255
38,
        0.72045861,  1.2299583 ,  0.37598859,  1.05729685,  0.368161
04,
        0.73800214, -0.52021731,  0.9113379 , -0.01292322, -0.150201
74,
        0.42437533,  0.52894687,  0.70241525, -0.05385675,  0.862776
68,
        0.33388091,  0.13504146,  1.19457128,  0.13677262,  0.727112
01,
        0.45564294,  1.21038365,  0.61327355,  0.68512164,  0.859514
24,
        1.23906722,  1.34575245,  0.64732915, -0.06728496,  0.100332
85,
       -0.00745518,  0.2179082 ,  1.04116321,  0.20719664,  0.421541
38,
        1.03691937,  1.015435  ,  0.0519502 ,  0.25582921,  0.253848
13,
        0.60915822,  0.31116522, -0.39679548,  0.26536661,  0.073858
97,
        0.20854936,  0.55843737, -0.47853403,  0.23545172,  1.840803
7 ,
        1.13831104,  2.19615974,  1.42613827,  1.8575403 ,  2.745111
73,
        0.34010583,  2.29180093,  1.98618025,  2.26382116,  1.355918
21,
        1.58471851,  1.87342402,  1.23656166,  1.45128483,  1.582762
83,
        1.45956552,  2.43560434,  3.29752602,  1.23377366,  2.032182
82,
        0.95980311,  2.88717988,  1.31405636,  1.69619746,  1.948687
73,
        1.1574572 ,  1.007133  ,  1.7733922 ,  1.85327106,  2.423478
8 ,
        2.31353522,  1.84800289,  1.09649923,  1.1812503 ,  2.791788
61,
        1.57340399,  1.33614369,  0.91061354,  1.84350913,  2.007011
61,
        1.89319854,  1.13831104,  2.03519535,  1.99464025,  1.859771
29,
        1.54200377,  1.50925493,  1.3690965 ,  0.94680339])
```

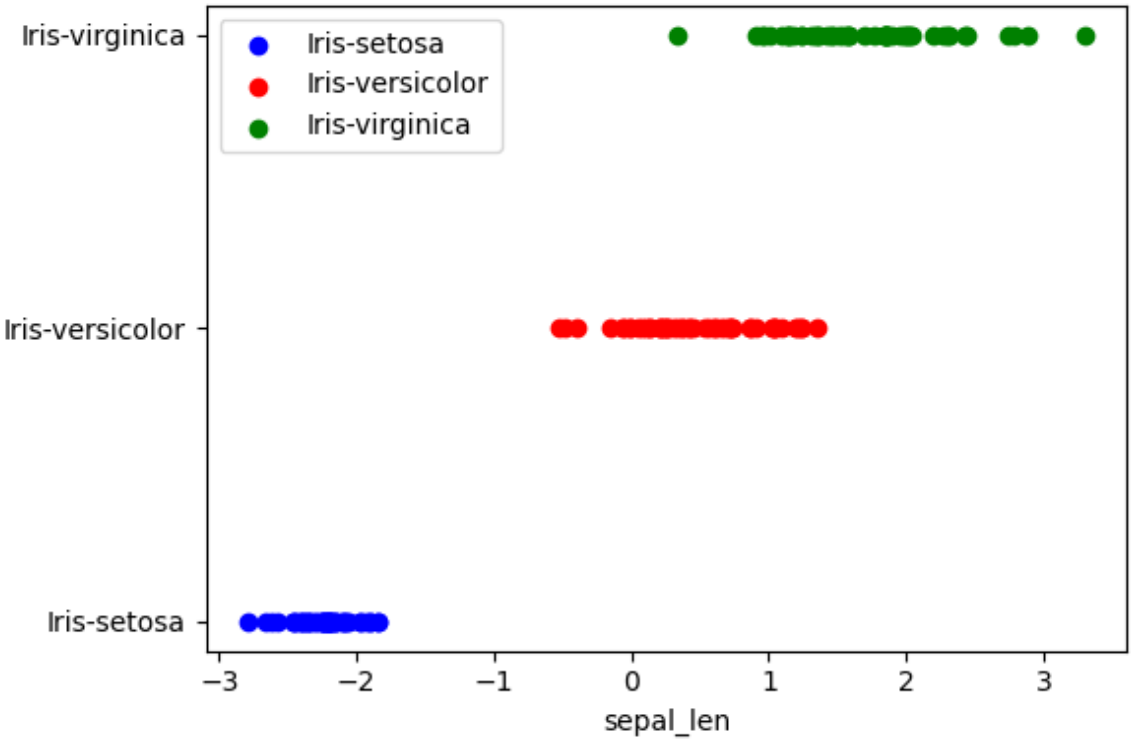
In [30]:

```
import matplotlib.pyplot as plt

# 创建 2x2 的子图布局
fig, axes = plt.subplots(2, 2, figsize=(10, 8))
feature_names = ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']
# 特征名称
class_names = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']  # 类别名称
colors = ['blue', 'red', 'green']

# 遍历四个特征，绘制 4 个子图
for i, ax in enumerate(axes.ravel()):  # ravel() 将 2x2 的 axes 摊平成1维
    for idx, (lab, col) in enumerate(zip(class_names, colors)):
        ax.scatter(X[y == lab, i],  # 当前特征 X[:, i]
                   [idx] * sum(y == lab),  # 用类别的索引作为 y 轴
                   label=lab,
                   color=col,
                   alpha=0.5)

    # 设置 x, y 轴标签和标题
    ax.set_xlabel(feature_names[i])  # 设置 x 轴为当前特征名称
    ax.set_yticks([0, 1, 2])  # 设置类别索引为 y 轴刻度
    ax.set_yticklabels(class_names)  # 设置 y 轴标签为类别名称
    ax.legend(loc='best')

# 调整布局和显示
plt.tight_layout()
plt.show()
```



In [27]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))
for idx, (lab, col) in enumerate(zip(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'),
                                     ('blue', 'red', 'green'))):
    plt.scatter(Y1[y==lab],  # 只使用第一个变量 sepal_len
                [idx] * sum(y == lab),  # 用类别的索引来表示 y 轴
                label=lab,
                c=col)

plt.xlabel('sepal_len')
plt.yticks([0, 1, 2], ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])  # 设置 y 轴刻度
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



In [ ]: