示例代码：

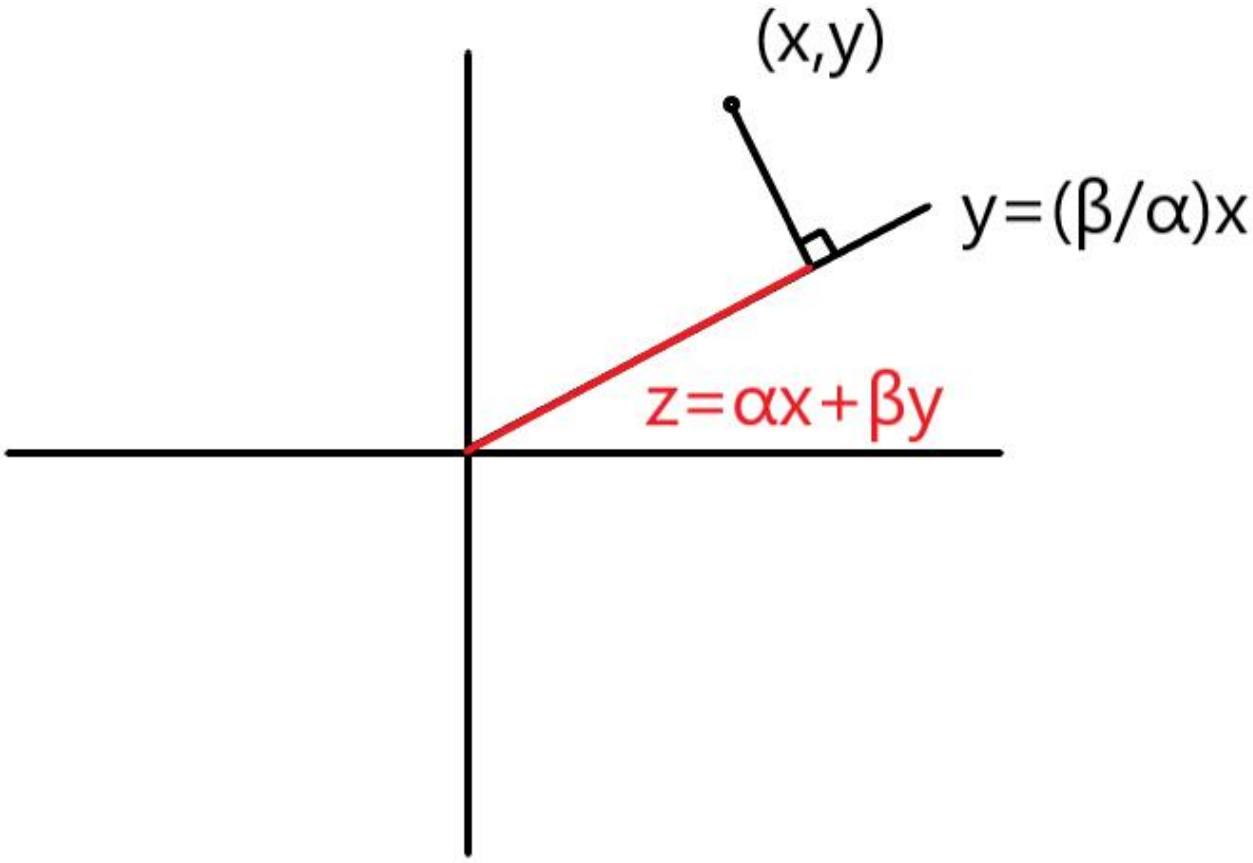
```
import numpy as np
from numpy import linalg
import matplotlib.pyplot as plt

class PCA:

    '''
    dataset 形如array([样本1,样本2,...,样本m]),每个样本是一个n维的ndarray
    '''
    def __init__(self, dataset):
        # 这里的参数跟上文是反着来的(每行是一个样本)，需要转置一下
        self.dataset = np.matrix(dataset, dtype='float64').T

    '''
    求主成分;
    threshold可选参数表示方差累计达到threshold后就不再取后面的特征向量.
    '''
    def principal_comps(self, threshold = 0.85):
        # 返回满足要求的特征向量
        ret = []
        data = []

        # 标准化
        for (index, line) in enumerate(self.dataset):
            self.dataset[index] -= np.mean(line)
            # np.std(line, ddof = 1)即样本标准差(分母为n - 1)
            #self.dataset[index] /= np.std(line, ddof = 1)
        # 求协方差矩阵
        Cov = np.cov(self.dataset)
        # 求特征值和特征向量
        eigs, vectors = linalg.eig(Cov)
        # 第i个特征向量是第i列，为了便于观察将其转置一下
        for i in range(len(eigs)):
            data.append((eigs[i], vectors[:, i].T))
        # 按照特征值从大到小排序
        data.sort(key = lambda x: x[0], reverse = True)

        sum = 0
        for comp in data:
            sum += comp[0] / np.sum(eigs)
            ret.append(
                tuple(map(
                    # 保留5位小数
                    lambda x: np.round(x, 5),
                    # 特征向量、方差贡献率、累计方差贡献率
                    (comp[1], comp[0] / np.sum(eigs), sum)
                ))
            )
            print('特征值:', comp[0], '特征向量:', ret[-1][0], '方差贡献率:',
ret[-1][1], '累计方差贡献率:', ret[-1][2])
```

```
            if sum > threshold:
                return ret

        return ret

# 生成干扰过的 y = x
def get_dataset():
    lst = []
    for i in range(30):
        x = np.random.random() * 10 # 0-10
        y = x + np.random.random()
        lst.append([x, y])
    return np.array(lst, dtype = 'float64')

dataset = get_dataset()
for point in dataset:
    plt.scatter(point[0], point[1])

pca = PCA(dataset)

# 返回的结果
comps = pca.principal_comps()

# [alpha, beta] 为第一个主成分的特征向量
[alpha, beta] = comps[0][0]

# 画变换后的直线
x = np.linspace(0, 10)
y = beta / alpha * x
plt.plot(x, y, c = 'b')

plt.show()
```

由于高维空间难以表示，仅以二维情况说明。 get_dataset方法返回一个干扰过了的y=x曲线，即在函数值后面随机偏移0~1. 采用主成分分析法降维，即构造原来变量x,y的线性组合z=αx+βy,使得变换后的样本的z值方差最大(方差最大表明数据分散程度大，因此可能会含更多的信息，更具体地可以看那篇博客的第二个参考资料)。 运行下面的代码

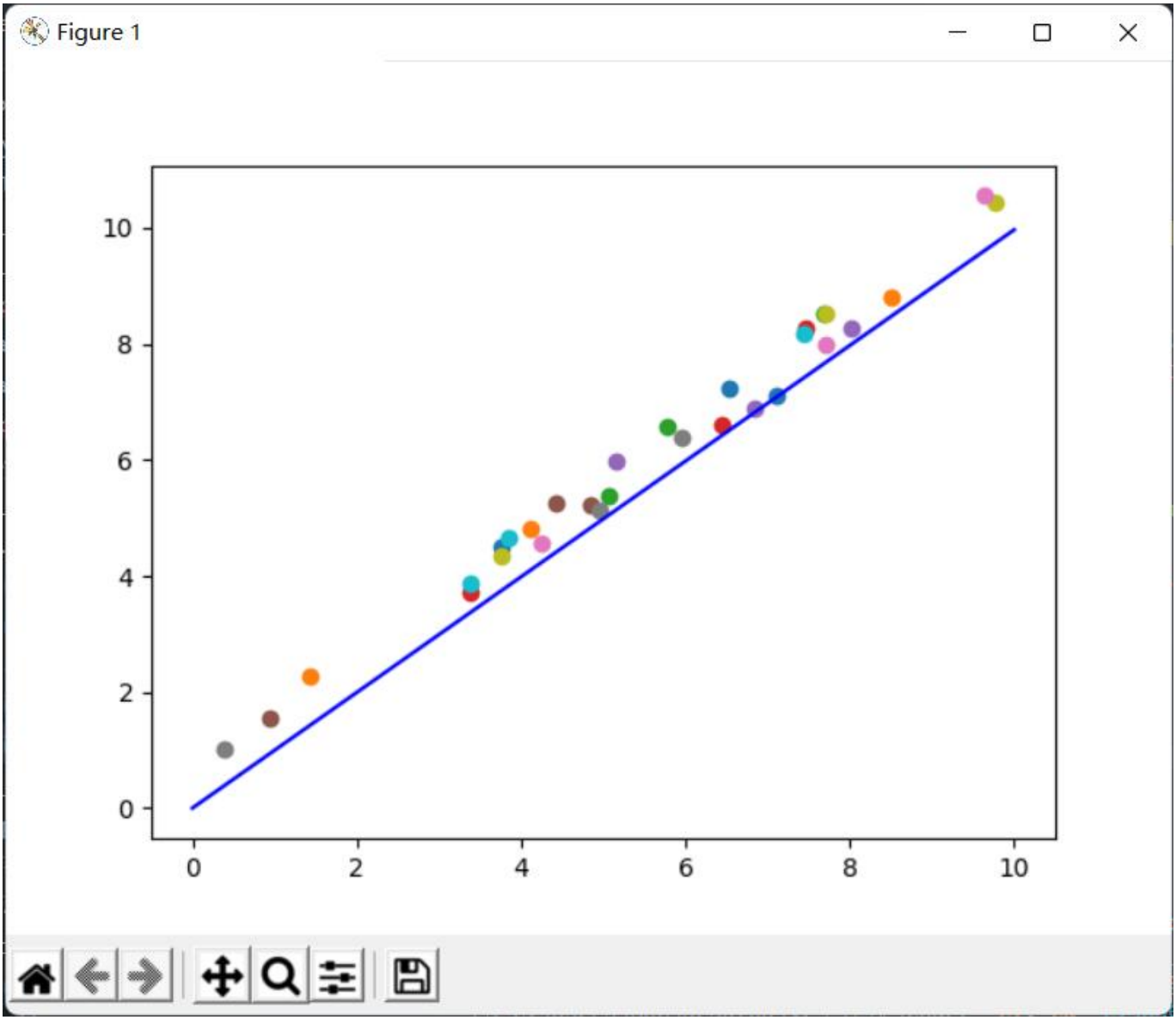
```
pca = PCA(dataset)

# 返回的结果
comps = pca.principal_comps()

# [alpha, beta] 为第一个主成分的特征向量
[alpha, beta] = comps[0][0]
```

可以得到第一个主成分z1的特征向量[α,β]为[0.70782, 0.70639](由于随机，每次运行应该是不一样的但是差距不大).将其视为坐标变换，可以在坐标系上画出y=(β/α)x,通过简单的计算可以证明，一个样本(x0,y0)经过坐标变换z=αx0+βy0得到的z值即其投影到直线上的点到原坐标系原点的距离。见下图：



返回的方差贡献率即主成分z1的方差(我运行的这次是0.99822)。

运行截图：



蓝线即为直线y=(β/α)x.