第5章

机器学习：无监督学习

无监督学习就是从无标注数据样本出发，学习数据样本中蕴含的模式，主要应用场景包括聚类分析(Cluster Analysis)、关联规则(Association Rule)、维度缩减(Dimensionality Reduce)等。在现实世界中，高质量、大规模有标注数据通常难以获得，作为监督学习的必要补充，通过无监督学习快速将无标注数据进行分类显得尤为重要。本章将从聚类、特征降维以及模型学习的角度介绍无监督学习，包括K-means聚类、主成分分析、特征脸方法、局部线性嵌入以及独立成分分析。

5.1 K-means聚类

K-means(K均值)原型最早由Stuart Lloyd于1957年提出，被用于脉码调制技术。而在1967年，James MacQueen发表了相关论文并正式提出了K-means聚类算法。通过聚类算法，将大量数据样本根据其特征相似性分为若干个簇，从而方便用户对数据进一步分析。K-means聚类算法要求特征变量连续且没有异常数据。它实现起来比较简单，聚类效果也不错，因此具有较为广泛的应用。

5.1.1 K-means聚类原理

K-means聚类算法解决的问题是，在没有标签数据的前提下，使算法根据距离的远近将n个数据最优地划分为k个类(簇)。它是无监督学习中比较常见的一种算法，原理简单易懂。本质是通过循环，不断迭代类中心点，计算各个数据到新的类中心点的距离并根据距离最近的原则重新归类，当类内距离最小、类间距离最大时，即可停止迭代(在使用过程中，常常会限定迭代次数，以防算法陷入死循环。当达到预先设定的循环次数或类中心点不再发生变化时，则停止迭代并得到最终聚类结果)。其中，K-means聚类算法得到的聚类结果很容易受初始值影响，为了达到上述“局部最优解”，可以利用不同的初始值重复几次，常用的初始化方法包括forgy以及random partition。

5.1.2 K-means聚类算法

假设待聚类样本集合为$\mathbf{D}=\{\boldsymbol{x}_j\}_{j=1}^{n}$，$\boldsymbol{x}_j\in\mathbf{R}^d$，K-means聚类算法的目标是将数据集划分为$k(k<n)$类，使得划分后的$k$个子集合满足类内的误差平方和最小。在Stuart Lloyd提出的经典K-means聚类算法中，采取迭代优化策略，有效

地求解目标函数的局部最优解。具体的算法流程如下。

1. 初始化聚类中心

初始化 k 个聚类中心 $\boldsymbol{c}=\{\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_k\}$，$\boldsymbol{c}_j\in\mathbf{R}^d(1\leqslant j\leqslant k)$，每个聚类中心 $\boldsymbol{c}_j$ 所在的集合记作 $\boldsymbol{G}_j$。

2. 对数据进行聚类

该步骤的目标是将每个待聚类数据放入唯一一个聚类集合中，首先通过设定的相似度/距离函数计算每个待聚类数据和 k 个聚类中心的距离。以最常见的欧氏距离为例，计算 $\boldsymbol{x}_i$ 和 $\boldsymbol{c}_j$ 之间的欧氏距离：

$$\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{c}_j)=\sqrt{\sum_{o=1}^{d}(x_{i,o}-c_{j,o})^2}\ (1\leqslant i\leqslant n,1\leqslant j\leqslant k)$$

将每个 $\boldsymbol{x}_i$ 归并到与其距离最短的聚类中心所在的聚类集合中，即 $\arg\min\limits_{\boldsymbol{c}_j\in C}\mathrm{dist}(\boldsymbol{x}_i,\boldsymbol{c}_j)$。

3. 更新聚类中心

根据聚类结果更新聚类中心，即根据每个聚类集合中所包含的数据，求均值得到该聚类集合新的中心：

$$\boldsymbol{c}_j=\frac{1}{|\boldsymbol{G}_j|}\sum_{\boldsymbol{x}_i\in\boldsymbol{G}_j}\boldsymbol{x}_i$$

由此可以看出，K-means 聚类的名称来源于此，而求取均值的操作简单高效，因此 K-means 聚类算法在大规模数据上可以得到更为高效的应用。

4. 迭代

迭代的过程即根据新的聚类中心，重复执行步骤 2 与步骤 3，在达到停止条件时终止迭代，从而得到最终的聚类结果。聚类迭代的停止条件判断并不唯一，通常使用的方法包括以下两种。

(1) 达到最大迭代次数。

(2) 聚类中心不再变化。

从另一个角度理解 K-means 聚类算法，最小化所有数据与聚类中心的距离等同于最小化每个类簇的方差。K-means 在迭代过程中需要不断减少簇内数据与聚类中心的欧氏距离，这相当于簇内数据方差最小化的目标函数：

$$\arg\min_{G}\sum_{i=1}^{k}\sum_{x\in\boldsymbol{G}_i}\|\boldsymbol{x}-\boldsymbol{c}_i\|^2=\arg\min_{G}\sum_{i=1}^{k}|\boldsymbol{G}_i|\,\mathrm{var}\boldsymbol{G}_i$$

其中，第 i 个聚类簇的方差为

$$\mathrm{var}(\boldsymbol{G}_i)=\frac{1}{|\boldsymbol{G}_i|}\sum_{\boldsymbol{x}\in\boldsymbol{G}_i}\|\boldsymbol{x}-\boldsymbol{c}_i\|^2$$

通常而言，聚类算法的目标是得到一个类内距离小(或称为类内相似性大)而类间距离大(或称为类间相似性小)的聚类结果。K-means 聚类就是通过最小化簇内方差来实现类内相似性最大化，即最小化每个簇内的数据方差从而使得最终聚类结果中的数据所呈现出来的差异性最小。

例 5.1 假设有 8 个点：(3,1)，(3,2)，(4,1)，(4,2)，(1,3)，(1,4)，(2,3)，(2,4)。使用 K-means 聚类算法对其进行聚类。设初始聚类中心分别为(0,4)和(3,3)。请写出详细的计算过程。

解：

第一步：列出数据表格，如表 5.1 所示。

表 5.1　数据表格

	a	b
x_1	3	1
x_2	3	2
x_3	4	1
x_4	4	2
x_5	1	3
x_6	1	4
x_7	2	3
x_8	2	4

第二步：初始聚类中心分别为 $c_1(0,4)$和 $c_2(3,3)$，计算各点到两中心的距离，如表 5.2 所示。

表 5.2　各点到两中心的距离

	$c_1(0,4)$	$c_2(3,3)$
$x_1(3,1)$	4.242	2√
$x_2(3,2)$	3.605	1√
$x_3(4,1)$	5	2.236√
$x_4(4,2)$	4.472	1.414√
$x_5(1,3)$	1.414√	2
$x_6(1,4)$	1√	2.236
$x_7(2,3)$	2.236	1√
$x_8(2,4)$	2	1.414√

第三步：根据表 5.2 分成两簇$\{x_1,x_2,x_3,x_4,x_7,x_8\},\{x_5,x_6\}$。重新计算新的聚类中心 c_3,c_4。并计算新的距离表，如表 5.3 所示。

$$c_3=\left(\frac{3+3+4+4+2+2}{6},\frac{1+2+1+2+3+4}{6}\right)=(3,2.167)$$

$$c_4=\left(\frac{1+1}{2},\frac{3+4}{2}\right)=(1,3.5)$$

表 5.3　新的距离表 1

	$c_3(3,2.167)$	$c_4(1,3.5)$
$x_1(3,1)$	1.167√	3.201

续表

	$c_3(3,2.167)$	$c_4(1,3.5)$
$x_2(3,2)$	0.167√	2.5
$x_3(4,1)$	1.536√	3.905
$x_4(4,2)$	1.013√	3.354
$x_5(1,3)$	2.166	0.5√
$x_6(1,4)$	2.712	0.5√
$x_7(2,3)$	1.301	1.118√
$x_8(2,4)$	2.088	1.118√

第四步：根据表 5.3 分成两簇$\{x_1,x_2,x_3,x_4\},\{x_5,x_6,x_7,x_8\}$。重新计算新的聚类中心 c_5,c_6。并计算新的距离表，如表 5.4 所示。

$$c_5=\left(\frac{3+3+4+4}{4},\frac{1+2+1+2}{4}\right)=(3.5,1.5)$$

$$c_6=\left(\frac{1+1+2+2}{4},\frac{3+4+3+4}{4}\right)=(1.5,3.5)$$

表 5.4 新的距离表 2

	$c_5(3.5,1.5)$	$c_6(1.5,3.5)$
$x_1(3,1)$	0.707√	2.915
$x_2(3,2)$	0.707√	2.121
$x_3(4,1)$	0.707√	3.535
$x_4(4,2)$	0.707√	2.915
$x_5(1,3)$	2.915	0.707√
$x_6(1,4)$	3.535	0.707√
$x_7(2,3)$	2.121	0.707√
$x_8(2,4)$	2.915	0.707√

第五步：根据表 5.4 分成两簇$\{x_1,x_2,x_3,x_4\},\{x_5,x_6,x_7,x_8\}$，和第四步分簇一致，停止计算。

5.1.3 K-means 聚类算法特点

1. 优点

由上述算法流程可以看出，K-means 聚类算法思想简单，容易理解。当数据分布接近高斯分布的时候，聚类效果非常不错。而对于大多数样本也可以获得较好的聚类效果，尽管是局部最优，但依然可以满足大部分任务。并且 K-means 聚类算法在处理大数据集的时候，可以保证较好的伸缩性。

此外，K-means 聚类算法收敛速度很快。首先，在样本分配阶段，需要计算 kn 次误差

平方和，计算复杂度为 $O(knd)$。其次，在更新聚类中心阶段，计算复杂度为 $O(nd)$。如果迭代次数为 t，则算法的计算复杂度为 $O(kndt)$。因此可以看出，K-means 聚类算法针对 n 个样本个数具有线性的计算复杂度，是一种非常高效的大数据聚类算法。

2. 缺点

K-means 聚类算法需要事先指定聚类数 k，不同 k 值得到的结果不一样，而很多时候并不知道数据应该被聚为多少类。通常使用的方法是遍历一个范围内的候选值然后测试错误率，但通常因为可选范围较大而并不切实际。

K-means 聚类算法对初始的聚类中心较为敏感，不同的选取方式对聚类的结果有较大影响。这是由于 K-means 聚类算法仅对目标函数求取近似局部最优解，不能保证得到全局最优解，即在一定数据分布下聚类结果会因为初始化的不同产生很大偏差。

K-means 聚类算法对异常数据非常敏感。K-means 聚类算法假设数据是没有离群点的，它对离群点的处理与其他数据一样，然而离群点对均值计算影响较大，会让聚类中心偏离没有离群点的中心，从而影响聚类效果。为此，可以使用目标函数或 K-medoids 算法来减小离群点对聚类结果的影响。K-medoids 算法选取的中心点(medoids)属于聚类簇中的一个点，这是与 K-means 聚类算法的主要区别。

每个样本只能归到某个固定类别，即 K-means 聚类算法对每个数据的归属判定非 1 即 0，不可能同属于多个类别，这种聚类方法被称为“硬聚类”。然而，由于数据或初始值的轻微改变，聚类边缘的数据点很容易被改变聚类类别，这些数据点可能有更好的归属判定方式。例如，高斯混合模型，通过概率的形式判定数据的归属，通过设置属于不同簇的概率来判断每个数据点的归属方式。

K-means 聚类算法对数据的尺度很敏感，也就是说，对数据所在的坐标空间敏感，例如，某个长度特征以 cm 还是以 m 为单位对最终的聚类结果有较大的影响，这是由于欧氏距离假设数据每个维度的重要性是一样的。

经典的 K-means 聚类算法采取二次欧氏距离作为相似性度量，并且假设目标函数的误差服从标准的正态分布，因此，K-means 聚类算法在处理非标准正态分布或非均匀分布的数据时效果较差。

5.1.4　K-means 聚类算法的改进

1. 聚类中心初始化的改进

在标准 K-means 聚类算法中，初始聚类中心使用随机采样的方式，不能保证得到的期望聚类结果。为了获得更好的聚类结果，可以多次随机初始化聚类中心，通过对比多组结果从而选择最优聚类结果。但是这样做会大大影响计算时间，那么如何更好地初始化聚类中心？

其中最有效简单的改进方式是 David Arthur 提出的 K-means＋＋聚类算法，该算法能够有效地产生初始的聚类中心，保证初始化后的 K-means 可以得到 $O(\log k)$ 的近似解。首先随机初始化一个聚类中心 $C=\{c_1\}$，然后通过迭代计算最大概率值：

$$x^* = \arg\max_x \frac{\mathrm{dist}(x,C)}{\sum_{j=1,2,\cdots,n} \mathrm{dist}(x_j,C)}$$

然后将 x^* 加入到下一个聚类中心：

$$C \leftarrow C \cup \{x^*\}$$

直到选择 k 个中心。

K-means＋＋聚类算法的计算复杂度为 $O(knd)$，没有增加过多的计算负担，同时可以保证算法更有效地接近于最优解。

2. 类别个数的自适应确定

聚类算法中的类别数对聚类效果有较大的影响，而该参数根据自身的先验知识或启发式来确定。例如，事先已经知道数据特征的大致分布或样本中包含的属性个数，如数字、性别等，那么如何在算法中加入自适应决定类别个数的过程？

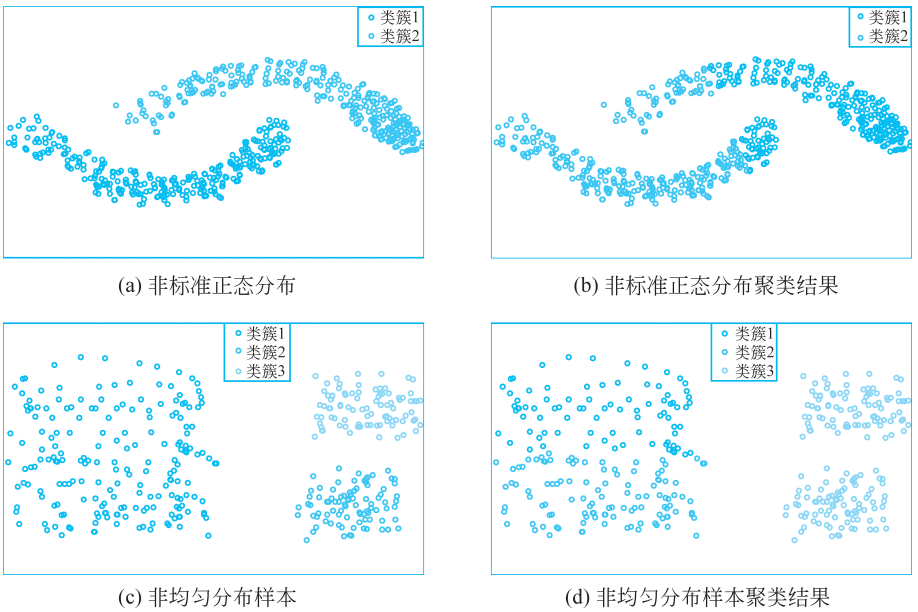
ISODATA 算法是经典的改进方法，该算法与 K-means 聚类算法在基本原则上一致，即通过计算误差平方和最小来实现聚类。但是 ISODATA 算法在迭代过程中引入类别的合并与分开机制。在每次迭代的过程中，ISODATA 算法首先在固定类别数的前提下进行聚类，然后根据设定样本之间的距离阈值进行合并操作，并根据每一组类别 G_i 中样本协方差矩阵信息来判断是否分开。

ISODATA 算法在 K-means 聚类算法的基础上引入了启发式重初始化，相比于经典的 K-means 聚类算法，计算效率大打折扣。

3. 面向非标准正态分布或非均匀分布数据的算法改进

如图 5.1 所示，针对非标准正态分布和非均匀分布的数据时，K-means 聚类算法不能得到预期结果，原因在于假设相似度度量为欧氏距离，而在实际数据集合中该假设不一定都会成立。

图 5.1 彩图



(a) 非标准正态分布　(b) 非标准正态分布聚类结果

(c) 非均匀分布样本　(d) 非均匀分布样本聚类结果

图 5.1　非标准正态分布（上）和非均匀分布样本（下）的聚类结果

为了克服该假设的局限性，K-means 聚类算法需要推广到更广义的度量空间。经典的两种改进框架为 Kernel K-means 和谱聚类 Spectral Clustering。

Kernel K-means 聚类算法将数据点 x_i 通过某种映射方式 $x_i \rightarrow \phi(x_i)$ 映射到新的高维空间 Φ，在该空间中数据点之间的内积可以通过对应的核函数进行计算，即：

$$k(x_i, x_j) = \phi(x_i)^{\mathrm{T}} \phi(x_j)$$

借助核函数，可以在新的高维空间对数据进行 K-means 聚类，样本之间的相似性度量就取决于核函数的选择。

谱聚类算法尝试着变换数据的度量空间，首先需要求取数据集合的仿射矩阵，然后计算仿射矩阵的特征向量，利用得到的特征向量进行 K-means 聚类。仿射矩阵的特征向量隐含地重新定义样本点的相似性。

4. 二分 K-means 聚类

按照 K-means 聚类规则很容易陷入局部最小值，为了克服这一问题，提出了二分 K-means 聚类算法。它首先将所有数据点作为一个簇，然后将该簇一分为二，之后选择其中一个簇继续进行划分，选择哪一个簇进行划分取决于对其划分是否可以最大程度降低误差平方和，不断重复上述划分过程，直到得到用户指定的聚类簇数目为止。

5.1.5　K-means 聚类算法的 Python 实现

```
import numpy as np
import random
import matplotlib.pyplot as plt

def distance(point1, point2):                    #计算距离(欧几里得距离)
    return np.sqrt(np.sum((point1 - point2) ** 2))

def k_means(data, k, max_iter=10000):
    centers = {}                                 #初始聚类中心
    #初始化,随机选 k 个样本作为初始聚类中心。random.sample(): 随机不重复抽取 k 个值
    n_data = data.shape[0]                       #样本个数
    for idx, i in enumerate(random.sample(range(n_data), k)):
        #idx 取值范围[0, k-1],代表第几个聚类中心;data[i]为随机选取的样本作为聚类
        #中心
        centers[idx] = data[i]

    #开始迭代
    for i in range(max_iter):                    #迭代次数
        print("开始第{}次迭代".format(i+1))
        clusters = {}                    #聚类结果,聚类中心的索引 idx -> [样本集合]
        for j in range(k):                       #初始化为空列表
            clusters[j] = []

        for sample in data:                      #遍历每个样本
            distances = []          #计算该样本到每个聚类中心的距离 (只会有 k 个元素)
            for c in centers:                    #遍历每个聚类中心
                #添加该样本点到聚类中心的距离
                distances.append(distance(sample, centers[c]))
```

```
            idx = np.argmin(distances)              #最小距离的索引
            clusters[idx].append(sample)            #将该样本添加到第 idx 个聚类中心

        pre_centers = centers.copy()                #记录之前的聚类中心点

        for c in clusters.keys():
            #重新计算中心点(计算该聚类中心的所有样本的均值)
            centers[c] = np.mean(clusters[c], axis=0)

        is_convergent = True
        for c in centers:
            if distance(pre_centers[c], centers[c]) > 1e-8:  #中心点是否变化
                is_convergent = False
                break
        if is_convergent == True:
            #如果新旧聚类中心不变,则迭代停止
            break
    return centers, clusters

def predict(p_data, centers):                       #预测新样本点所在的类
    #计算 p_data 到每个聚类中心的距离,然后返回距离最小所在的聚类
    distances = [distance(p_data, centers[c]) for c in centers]
    return np.argmin(distances)
```

5.2 主成分分析

在一些应用场景中，需要处理的数据的特征维度非常高。以图像为例，对于高和宽都为100px的图像，如果将所有像素拼接起来成为一个向量，这个向量的维度为10 000。一般情况下，数据的各个指标之间存在着一定的相关性，从而增加了问题分析的复杂性，并且直接利用这些指标构建机器学习模型会显著降低运算效率。如果分别对数据的每个指标进行分析，由于分析通常是孤立的，不能完全利用数据中的信息，因此盲目减少指标会损失较多有用的信息，从而产生错误的结论。

因此，在减少需要分析的指标同时，尽量减少原指标所包含信息的损失，使得样本数据能够达到全面分析问题的目的。由于各变量之间存在一定的相关关系，因此可以将关系紧密的变量变成尽可能少的新变量，使这些新变量是两两不相关的，那么就可以用较少的综合指标分别代表存在于各个变量中的各类信息。主成分分析(Principal Component Analysis，PCA)就是达到这种目的的方法之一。

5.2.1 主成分分析原理

主成分分析属于典型的数据降维方法，降维是一种对高维度数据进行预处理的方法。即将高维度数据保留下最重要的一些特征，去除数据噪声与不重要的特征，从而实现提升数据处理速度的目的。例如，对于图像数据，要求保持视觉对象区域构成的空间分布；对于文本数据，要求保持文本之间的(共现)相似或不相似的特性。在实际的生产和应用中，降维在

一定的信息损失范围内，可以节省大量的时间和成本。在介绍主成分分析之前，先回顾相关的数学知识，包括方差、协方差、相关系数。

1. 方差

方差描述了样本数据的波动程度，当数据分布比较分散（即数据在平均值附近波动比较大）时，方差就大，反之则小。方差等于各个数据与样本均值之差的平方和的平均数，假设有 n 个数据，记为 $X=\{x_i\}(i=1,2,\cdots,n)$，那么样本方差（sample variance）为

$$\mathrm{var}(X)=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-u)^2$$

其中，u 是样本均值，$u=(\sum_{i=1}^{n}x_i)/n$。上述样本方差公式里分母为 $n-1$ 的目的是让对方差的估计是无偏估计。

2. 协方差

协方差用于衡量两个变量之间的相关度，方差是协方差的一种特例，即当两个变量是相同情况。假设有两个变量，观察到不同时刻两个变量的取值，记作 $(X,Y)=\{(x_i,y_i)\mid i=1,\cdots,n\}$，那么两个变量的协方差为

$$\mathrm{cov}(X,Y)=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-E(X))(y_i-E(Y))$$

其中，$E(X)$与 $E(Y)$分别表示 X 和 Y 的样本均值，分别定义为

$$E(X)=\frac{1}{n}\sum_{i=1}^{n}x_i,E(Y)=\frac{1}{n}\sum_{i=1}^{n}y_i$$

表 5.5 给出了一组身高（X）与体重（Y）方差与协方差的例子。

表 5.5　方差与协方差的计算例子（无偏估计）

	身高 X/cm	体重 Y/500g	$X-E(X)$	$Y-E(Y)$	$[X-E(X)][Y-E(Y)]$
1	152	92	−19.4	−39.7	770.2
2	185	162	13.6	30.3	412.1
3	169	125	−2.4	−6.7	16.1
4	172	118	0.6	−13.7	−8.2
5	174	122	2.6	−9.7	−25.2
6	168	135	−3.4	3.3	−11.2
7	180	168	8.6	36.3	312.2
	$E(X)=171.4$	$E(Y)=131.7$	$\mathrm{var}(X)=94.2$	$\mathrm{var}(Y)=592.8$	$\mathrm{cov}(X,Y)=209.4$

对于一组二维变量（例如，身高-体重、广告投入-商品销售、天气状况-旅行计划等），可通过计算它们之间的协方差值来判断这组数据给出的二维变量是否存在关联关系。

当协方差 $\mathrm{cov}(X,Y)>0$ 时，则 X 与 Y 正相关。

当协方差 $\mathrm{cov}(X,Y)<0$ 时，则 X 与 Y 负相关。

当协方差 $\mathrm{cov}(X,Y)=0$ 时，则 X 与 Y 不相关。

从表 5.5 中可以看出，身高较高的体重一般会比较大，同样，体重大的身高一般也比较

高，得到的结果也是正相关的，计算出来的结果也非常符合直觉认知。

3. 相关系数

由于协方差计算的结果会受到变量取值尺度的影响，而皮尔逊相关系数（Pearson Correlation Coefficient）通过将两组变量之间的关联度规范到一定取值范围内很好地解决了这个问题。它的定义方式如下：

$$\mathrm{corr}(X,Y)=\frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}}=\frac{\mathrm{cov}(X,Y)}{\sigma_x\sigma_y}$$

其中，σ_x 与 σ_y 分别表示 X 与 Y 的标准差。

根据施瓦茨不等式可以得到皮尔逊相关系数的性质如下：

$$|\mathrm{corr}(X,Y)|\leqslant 1$$

当 $\mathrm{corr}(X,Y)=1$ 时，说明两个随机变量完全正相关，即满足 $Y=aX+b$，$a>0$。考虑 $\mathrm{corr}(X,X)$，两个随机变量相同，一定满足线性关系，此时，$\mathrm{cov}(X,X)=\mathrm{var}(X)$，容易得到 $\mathrm{corr}(X,Y)=1$。当 $\mathrm{corr}(X,Y)=-1$ 时，说明两个随机变量完全负相关，即满足 $Y=-aX+b$，$a>0$。而当 $0<|\mathrm{corr}(X,Y)|<1$ 时，说明两个随机变量之间存在一定的线性关系。其中，正相关即表示变量 X 增加的情况下，变量 Y 也随之增加；而负相关表示变量 X 减少的情况下，变量 Y 随之增加。以表 5.5 为例，身高体重的相关系数为 $\mathrm{corr}(X,Y)=209.4/(9.7\times 24.3)=0.888$。

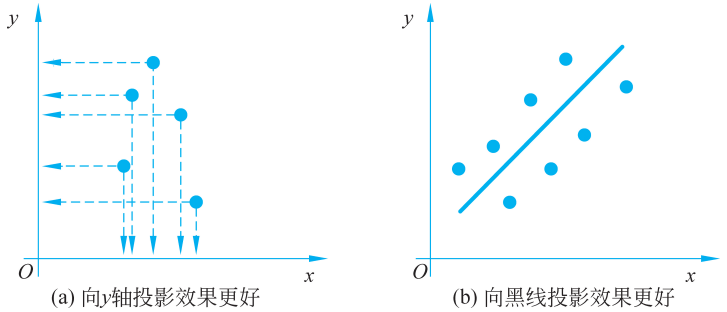
从上述定义可以看出，皮尔逊相关系数具有对称的属性，即：

$$\mathrm{corr}(X,Y)=\mathrm{corr}(Y,X)$$

由上述定义可以得出如下性质：皮尔逊相关系数描述了两个变量的线性相关程度，如果 $|\mathrm{corr}(X,Y)|$ 的取值越大，则两者之间存在相关程度较大的线性关系。而当 $|\mathrm{corr}(X,Y)|=0$ 时，表示两者之间不存在线性关系，但不是没有关系，可能存在其他非线性关系。

4. 主成分分析

回到特征降维，降维需要尽可能将数据向方差最大的方向进行投影，使得数据所包含的信息丢失尽可能少，且保留最主要的特征成分，从而增加特征的判别性。如图 5.2(a)所示，这些二维数据向一维空间投影时，向 y 方向投影明显比向 x 方向投影的降维意义更大，因为在 y 方向的特征具有更大的方差，从而使得特征具有更强的判别性；而图 5.2(b)则往黑线方向投影效果更好。这些投影方式更好地保留了未降维之前数据的离散程度。



(a) 向y轴投影效果更好　　(b) 向黑线投影效果更好

图 5.2　PCA 降维示意图

主成分分析的思想是将 d 维特征数据映射到 l 维空间（一般 $d\gg l$），这 l 维是全新的正交特征也被称为主成分，是在原始 d 维特征数据的基础上重新构造的 l 维特征。主成分分