第3章

认识数据

数据集由数据对象组成，每个数据对象表示一个实体。例如，在选课数据库中，对象可以是教师、课程和学生；在医疗数据库中，对象可以是患者。数据对象又称为实例、样本或对象。如果数据对象存放在数据库中，则它们称为元组。一般数据库的行对应于数据对象，而列对应于属性。

视频讲解

3.1 属性及其类型

3.1.1 属性

属性(Attribute)是一个数据字段，表示数据对象的一个特征。在文献中，属性、维(Dimension)、特征(Feature)和变量(Variable)表示相同的含义，可以在不同场合互换使用。术语“维”一般用于数据仓库中；“特征”较多地用于机器学习领域；统计学中更多地使用“变量”；数据库和数据仓库领域一般使用“属性”或“字段”。例如，描述学生的属性可能包括学号、姓名、性别等。通常把描述对象的一组属性称作该对象的属性向量。

3.1.2 属性类型

属性的取值范围决定了属性的类型。属性类型一般分为两大类，一类是定性描述的属性，一般有标称属性、二元属性和序数属性；另一类是定量描述的属性，即数值属性，一般可以是整数或连续值。数据对象的属性类型划分如图 3-1 所示。

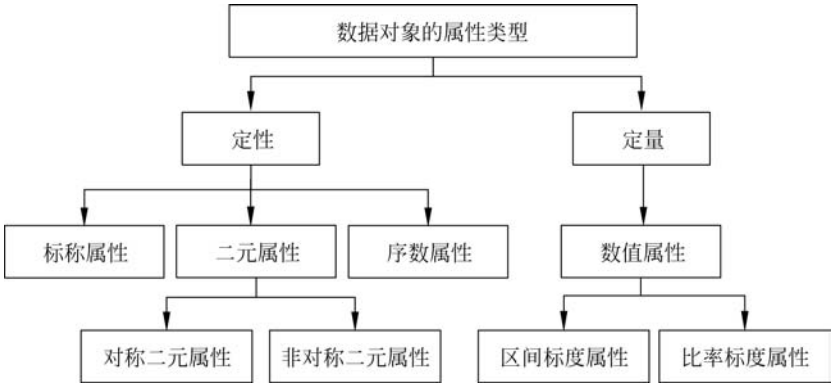

图 3-1　数据对象的属性类型

1. 标称属性

标称属性(Nominal Attribute)的值是一些符号或事物的名称。每个值代表某种类别、编码或状态,因此标称属性又可称为是分类的(Categorical)。标称属性的值是枚举的,可以用数字表示这些符号或名称。常见的标称属性有姓名、籍贯、邮政编码和婚姻状态等。标称属性的值不仅仅是不同的名字,它提供了足够的信息用于区分对象。鉴于标称属性值并不具有有意义的序,统计它的中位数和均值是没有意义的,但是可以找出某个出现次数最多的值。例如,出现次数最多的姓名,这个就可以用众数表示。因此,标称属性的中心趋势度量一般是众数。

2. 二元属性

二元属性(Binary Attribute)是标称属性的特例,也是一种布尔属性,对应 0 和 1 两个状态,分别表示 False 和 True。常见的二元属性有抛一枚硬币是正面朝上还是反面朝上、患者的检查结果是阴性还是阳性等。二元属性分为对称的和非对称的。如果属性的状态结果是同等重要的,如抛硬币的结果状态,则该属性是对称的二元属性。一个非对称的二元属性,其状态的结果不是同样重要的,如病毒检测的阳性和阴性结果,为了方便,用 1 对重要结果(通常是稀有的)编码,另一个用 0 编码。由于二元属性也是标称属性的一种,因此只能用众数统计二元属性。

3. 序数属性

序数属性(Ordinal Attribute)的可能值之间存在有意义的序或秩评定,但是相继值之间的差是未知的,常见的序数属性,如衣服的尺寸有 S、M、L、XL,可以用数字 1、2、3、4 分别对应属性的取值。由于序数属性是有序的,它的中位数是有意义的,因此序数属性的中心趋势度量可以是众数和中位数。

标称属性、二元属性和序数属性都是定性的。它们描述对象的特征而不给出实际大小或数量。这种定性属性的值通常是代表类别的词。如果使用整数描述,仅代表类别的

编码，而不是可测量的值。

4. 数值属性

数值属性(Numeric Attribute)是可以度量的量，用整数或实数值表示，常见的数值属性有年龄等。数值属性可以是区间标度的或比率标度的。区分区间标度和比率标度属性的原则是该属性是否有固有的零点，如摄氏温度没有固有的零点，比值没有意义，所以是区间标度属性；而开式温度有固有的零点，比值有意义，所以是比率标度属性。数值属性的均值是有意义的，如某个城市人口的平均年龄可以看出这个城市的老龄化情况，因此，数值属性可以用众数、中位数、均值 3 个中心趋势度量进行统计。

数值属性是定量的，可以是离散的，也可以是连续的。标称属性、二元属性、序数属性都是定性的，且都是离散的。


视频讲解

3.2 数据的基本统计描述

把握数据的分布对于成功的数据预处理是至关重要的。基本的数据统计描述可以识别数据的性质，并判断哪些数据是噪声或离群点。

数据的描述性统计主要包括数据的集中趋势、离中趋势、相对离散程度和分布的形状 4 个方面。

3.2.1 中心趋势度量

中心趋势在统计学中是指一组数据向某一中心值靠拢的程度，它反映了一组数据中心点的位置所在。中心趋势度量就是寻找数据水平的代表值或中心值。中心趋势度量包括均值、中位数、众数和中列数。

1. 均值

数据集“中心”的最常用的数值度量是(算术)均值。设某属性 X 的 N 个观测值为 $x_1,x_2,\cdots,x_N$，则该集合的均值(Mean)为

$$\bar{x}=\frac{\sum_{i=1}^{N}x_i}{N}=\frac{x_1+x_2+\cdots+x_N}{N} \tag{3.1}$$

在实际问题中，每个 x_i 可以与一个权重 ω_i 关联。权重反映它们所依附的对应值的重要性或出现的频率。当各项权重不相等时，计算均值时就要采用加权均值(Weighed Mean)，即

$$\bar{x}=\frac{\sum_{i=1}^{N}\omega_i x_i}{\sum_{i=1}^{N}\omega_i}=\frac{\omega_1x_1+\omega_2x_2+\cdots+\omega_Nx_N}{\omega_1+\omega_2+\cdots+\omega_N} \tag{3.2}$$

加权均值的大小不仅取决于总体中各单位的数值的大小，而且取决于各数值出现的次数(频数)。

均值是描述数据集的最常用统计量，但它并非度量数据中心的最佳方法，主要原因是均值对噪声数据很敏感。例如，一个班的某门课考试成绩的均值可能会被个别极低的分数拉低，或者某单位职工的平均工资会被个别高收入的工资抬高。为了减小少数极端值的影响，可以使用截尾均值(Trimmed Mean)。截尾均值是丢弃高低极端值后的均值，如对一幅图像的像素值可以按由小到大排列后去掉前后各 2%。截尾均值要避免在两端去除太多数据，否则会丢失有价值的信息。

2. 中位数

中位数(Median)又称为中点数或中值。中位数是按顺序排列的一组数据中居于中间位置的数，即在这组数据中，有一半的数据比它大，另一半的数据比它小。在概率论与统计学中，中位数一般用于数值型数据。在数据挖掘中可以把中位数推广到序数型数据中。假定有某属性 X 的 N 个值按递增顺序排列，如果 N 是奇数，则中位数是该有序数列的中间值；如果 N 是偶数，则中位数是中间两个值的任意一个。对于数值型数据，一般约定中位数取中间两个数的平均值。

当数据量很大时，中位数的计算开销会很大，此时可以采用近似估计的方法。假定数据可以根据数值划分为区间，并且知道每个区间的数据个数，可以使用如下公式计算中位数。

$$\text{median} = L_1 + \left[\frac{\frac{N}{2} - \left(\sum f\right)_l}{f_{\text{median}}}\right] \text{width} \tag{3.3}$$

其中，L_1 为中位数区间的下界；N 为数据集中的数据个数；$\left(\sum f\right)_l$ 为低于中位数区间的所有区间频率和；f_{median} 为中位数区间的频率；width 为中位数区间的宽度。

【例 3-1】 某企业 50 名工人加工零件的数据如表 3-1 所示，计算加工零件数值的中位数。

表 3-1 加工零件数统计数据

按加工零件数目分组/个	频数/人	按加工零件数目分组/个	频数/人
105～110	3	125～130	10
110～115	5	130～135	6
115～120	8	135～140	4
120～125	14		

由表 3-1 中数据可知，中位数的位置为 50/2＝25，即中位数在 120～125 这一组，由此可以得到 L_1 ＝120，$\left(\sum f\right)_l$ ＝16，f_{median} ＝14，width＝5，则近似计算的中位数 median 为 123.31。

3. 众数

众数(Mode)是一组数据中出现次数最多的数值。可以对定性和定量型属性确定众

数。有时众数在一组数中有好几个。具有一个、两个或 3 个众数的数据集分别称为单峰(Unimodal)、双峰(Bimodal)和三峰(Trimodal)。一般具有两个或以上众数的数据集是多峰的(Multimodal)。在极端情况下,如果每个数值只出现一次,则它没有众数。

对于非对称的单峰型数据集,一般有下面的经验关系。

$$\text{mean} - \text{mode} \approx 3 \times (\text{mean} - \text{median}) \tag{3.4}$$

4. 中列数

中列数(Midrange)是数据集中的最大值和最小值的均值,也可以度量数值数据的中心趋势。

【例 3-2】 利用 Pandas 统计中位数、均值和众数。

```
In[97]:    import pandas as pd
           df = pd.DataFrame([[1, 2], [7, -4],[3, 9], [4, -4],[1,3]],
           columns = ['one', 'two'])
           print('中位数: \n',df.median())
           print('均值: \n',df.mean(axis = 1))
           print('众数: \n',df.mode())
Out[97]:
           中位数:
           one    3.0
           two    2.0
           均值:
           0    1.5
           1    1.5
           2    6.0
           3    0.0
           4    2.0
           众数:
               one  two
           0    1   -4
```

3.2.2 数据散布度量

数据散布度量用于评估数值数据散布或发散的程度。散布度量的测定是对统计资料分散状况的测定,即找出各个变量值与集中趋势的偏离程度。通过度量散布趋势,可以清楚地了解一组变量值的分布情况。离散统计量越大,表示变量值与集中统计量的偏差越大,这组变量就越分散。这时,如果用集中量数去做估计,所出现的误差就较大。因此,散布趋势可以看作是中心趋势的补充说明。数据散布度量包括极差、分位数、四分位数、百分位数和四分位数极差。方差和标准差也可以描述数据分布的散布。

1. 极差、四分位数和四分位数极差

极差(Range)又称为范围误差或全距,是一组观测值的最大值与最小值之间的差距。

极差是标志值变动的最大范围，它是测定标志变动的最简单的指标。

分位数又称为分位点，是指将一个随机变量的概率分布范围分为几个等份的数值点，常用的有中位数（即二分位数）、四分位数和百分位数等。

四分位数是将一组数据由小到大（或由大到小）排序后，用 3 个点将全部数据分为 4 等份，与这 3 个点位置上相对应的数值称为四分位数，分别记为 Q_1（第一四分位数，说明数据中有 25%的数据小于或等于 Q_1）、Q_2（第二四分位数，即中位数，说明数据中有 50%的数据小于或等于 Q_2）和 Q_3（第三四分位数，说明数据中有 75%的数据小于或等于 Q_3）。其中，Q_3 到 Q_1 之间的距离的差的一半又称为分半四分位差，记为$(Q_3-Q_1)/2$。

第一四分位数和第三四分位数之间的距离是散布的一种简单度量，它给出被数据的中间一半所覆盖的范围。该距离称为四分位数极差（IQR），定义为

$$\mathrm{IQR}=Q_3-Q_1 \tag{3.5}$$

四分位数是把排序的数据集划分为 4 个相等的部分的 3 个值，假设有 12 个观测值且已经排序，则该数据集的四分位数分别是该有序表的第 3、第 6 和第 9 个数，四分位数极差是第 9 个数减去第 3 个数。

【例 3-3】 统计数据的分位数等统计量。

```
In[98]:    import pandas as pd
           df = pd.DataFrame([[1, 2], [7, -4],[3, 9], [3, -4]],
               index=['a', 'b', 'c', 'd'],columns=['one', 'two'])
           display(df)
           df.describe()
```

输出结果如图 3-2 所示。

2. 五数概括、盒图与离群点

五数概括法即用 5 个数概括数据，分别为最小值、第一四分位数（Q_1）、中位数（Q_2）、第三四分位数（Q_3）和最大值。

盒图（Boxplot）又称为盒式图或箱线图，是一种用作显示一组数据分散情况资料的统计图，因形状如箱子而得名。盒图体现了五数概括。

（1）盒图的边界分别为第一四分位数和第三四分位数；

（2）在箱体上中位数即第二四分数处作垂线；

（3）虚线称为触须线，触须线的端点为最小值和最大值。

利用四分位数间距 $\mathrm{IQR}=Q_3-Q_1$，找到界限，超出即为异常值。

$$\mathrm{IQR}_{左}=Q_1-1.5\times\mathrm{IQR}$$

$$\mathrm{IQR}_{右}=Q_3+1.5\times\mathrm{IQR}$$

每个异常值的位置用符号标出。箱线图提供了另一种检测异常值的方法，但它和 Z-分数检测出的异常值不一定相

	one	two
a	1	2
b	7	-4
c	3	9
d	3	-4

	one	two
count	4.000000	4.000000
mean	3.500000	0.750000
std	2.516611	6.184658
min	1.000000	-4.000000
25%	2.500000	-4.000000
50%	3.000000	-1.000000
75%	4.000000	3.750000
max	7.000000	9.000000

图 3-2 数据统计量

同，可选一种或两种。

如数据集的第一四分位数为 42，第三四分位数为 50，计算箱线图的上、下界限，并判断数据值 65 是否应该认为是一个异常值。

箱线图上限为 50＋1.5×8＝62，由于 65 大于上限，可以判定是异常值。

【例 3-4】 利用 Matplotlib 绘制箱线图。

```
In[99]:    import numpy as np
           import matplotlib.pyplot as plt
           import pandas as pd
           np.random.seed(2)                    #设置随机种子
           df = pd.DataFrame(np.random.rand(5,4),
           columns = ['A', 'B', 'C', 'D'])
           #生成 0～1 的 5×4 维度数据并存入 4 列 DataFrame 中
           plt.boxplot(df)
           plt.show()
```

输出结果如图 3-3 所示。

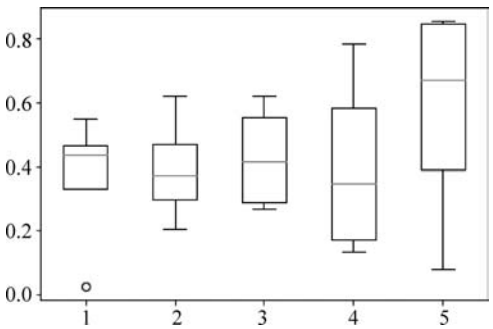

图 3-3　绘制箱线图

3.3　数据可视化

视频讲解

通过图形清晰有效地表达数据称为数据可视化（Data Visualization）。它将数据所包含的信息的综合体，包括属性和变量，抽象化为一些图表形式。有效的可视化能进一步帮助用户分析数据、推论事件和寻找规律，使复杂数据更容易被用户所理解和使用。

3.3.1　基于像素的可视化技术

基于像素的可视化方法是将对象的每一个数据属性映射到有限的屏幕空间内的一个像素点上，从而可视化尽可能多的数据对象，并且通过排列像素点体现出数据中所存在的模式。近年来，基于像素的可视化技术在很多具体场景中得到了广泛的应用，并且充分验证了方法的有效性。

3.3.2 几何投影可视化技术

几何投影技术可以帮助用户发现多维数据集的有趣投影。几何投影技术的难点在于在二维显示上可视化高维空间。散点图使用笛卡尔坐标显示二维数据点。使用不同颜色或形状表示不同的数据点,可以增加第三维。

【例 3-5】 Python 绘制散点图示例。

```
In[100]:    import matplotlib.pyplot as plt
            import numpy as np
            n = 50
            # 随机产生 50 个 0～2 的 x,y 坐标
            x = np.random.rand(n) * 2
            y = np.random.rand(n) * 2
            colors = np.random.rand(n)
            # 随机产生 50 个 0～1 的颜色值
            area = np.pi * (10 * np.random.rand(n)) ** 22
            # 点的半径范围:0～10
            plt.scatter(x, y, s = area, c = colors, alpha = 0.5, marker = (9, 3, 30))
            plt.show()
```

输出结果如图 3-4 所示。

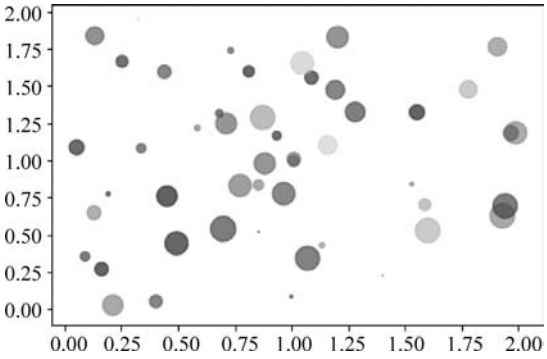

图 3-4 三维散点图

三维散点图使用笛卡尔坐标系的 3 个坐标轴。如果使用颜色信息,可以显示四维数据点。但对于超过四维的数据集,散点图一般不太有效。散点图矩阵是散点图的一种扩充,提供每个维与其他维的可视化。

【例 3-6】 Python 绘制散点图矩阵示例。

```
In[101]:    import seaborn as sns
            df_iris = sns.load_dataset('iris')
            sns.set(style = "ticks")
            g = sns.pairplot(df_iris,vars = ['sepal_length', 'petal_length'])
```

输出结果如图 3-5 所示。

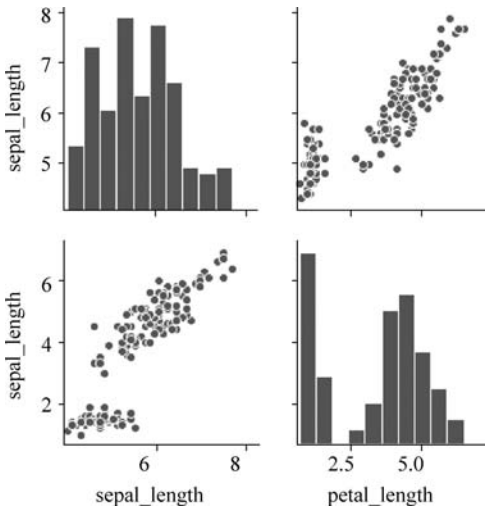

图 3-5　散点图矩阵

随着维度的增加，散点图矩阵变得不太有效。平行坐标图（Parallel Coordinates Plot）是对于具有多个属性问题的一种可视化方法。在平行坐标图中，数据集的一行数据在平行坐标图中用一条折线表示，纵向是属性值，横向是属性类别（用索引表示）。

【例 3-7】 Python 绘制平行坐标图示例。

```
In[102]:    from pyecharts.charts import Parallel
            import pyecharts.options as opts
            import seaborn as sns
            import numpy as np
            data = sns.load_dataset('iris')
            data_1 = np.array(data[['sepal_length', 'sepal_width', 'petal_length',
            'petal_width']]).tolist()
            parallel_axis = [
                {"dim": 0, "name": "萼片长度"},
                {"dim": 1, "name": "萼片宽度"},
                {"dim": 2, "name": "花瓣长度"},
                {"dim": 3, "name": "花瓣宽度"},
            ]
            parallel = Parallel(init_opts = opts.InitOpts(width = "600px", height = "400px"))
            parallel.add_schema(schema = parallel_axis)
            # parallel.config(schema)
            parallel.add('iris 平行图',data = data_1,linestyle_opts = opts.LineStyleOpts
            (width = 4, opacity = 0.5))
            parallel.render_notebook()
```

输出结果如图 3-6 所示。

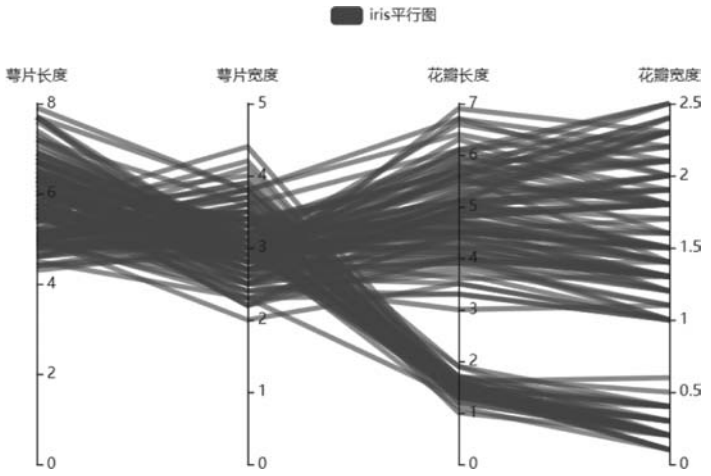

图 3-6　平行坐标图

3.3.3　基于图符的可视化技术

基于图符的(Icon-based)可视化技术使用少量图符表示多维数据值。有两种流行的基于图符的技术,即切尔诺夫脸和人物线条图。

1. 切尔诺夫脸

切尔诺夫脸(Chernoff Faces)是统计学家赫尔曼·切尔诺夫于 1973 年提出的,如图 3-7 所示。切尔诺夫脸把多达 18 个变量的多维数据通过卡通人物的脸显示出来,有助于揭示数据中的趋势。脸的要素有眼、耳、口和鼻等,用其形状、大小、位置和方向表示维度的值。切尔诺夫脸利用人的思维能力,识别面部特征的微小差异并且立即消化、理解许多面部特征。观察大型数据表可能是令人乏味的,切尔诺夫脸可以浓缩数据,从而更容易被消化理解,有助于数据的可视化。切尔诺夫脸有对称的切尔诺夫脸(18 维)和非对称的切尔诺夫脸(36 维)两种类型。

图 3-7　切尔诺夫脸(每张脸表示一个 N 维数据)

2. 人物线条图

人物线条画(Stick Figure)可视化技术把多维数据映射到 5 段人物线条画中,其中每幅画都有一个四肢和一个躯体。两个维度被映射到显示轴(x 轴和 y 轴),而其余的被映射到四肢角度和长度。图 3-8 显示的是人口普查数据,其中 Age 和 Income 被映射到显示轴,而其他维被映射到人物线条画。如果数据项关于两个显示维相对稠密,则结果可

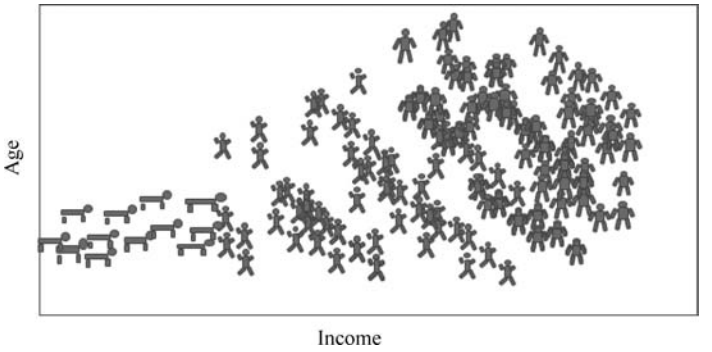

图 3-8　人物线条图

视化显示纹理模式，从而反映数据趋势。

3.3.4　层次可视化技术

对于大型高维数据集很难实现可视化技术。层次可视化把大型的高维数据的所有维划分成子集（子空间），在这些子空间按层次可视化。

3.3.5　可视化复杂对象和关系

早期的可视化技术主要用于分析数值数据，然而现在出现了越来越多的非数值数据，如文本和社会网络数据，可视化这些非数值数据引起了更多广泛的关注。标签云是一种用户产生的标签统计量的可视化。在标签云中，标签通常按字母次序或用户指定的次序列举。标签云的方法主要有两种。

（1）对于单个术语的标签云，根据不同用户使用该标签的次数显示该标签的大小。

（2）在多个术语上可视化标签统计量时，该标签被使用得越多，就使它显示得越大。

除了复杂的数据之外，数据项之间的复杂关系也对可视化提出了挑战。例如，使用疾病影响图可视化疾病之间的相关性，图中的节点代表疾病，节点的大小与对应疾病的流行程度成正比，如果对应疾病具有强相关性，则两个节点可以用一条边来连接，边的宽度与对应的疾病相关强度成正比。

3.3.6　高维数据可视化

无论是在日常生活中还是在科学研究中，高维数据处处可见。例如，一件简单的商品就包含了型号、厂家、价格、性能和售后服务等多种属性；再如，为了找到与致癌相关的基因，需要分析不同病人成百上千的基因表达。一般很难直观快速理解三维以上的数据，而将数据转化为可视的形式，可以帮助理解和分析高维空间中的数据特性。高维数据可视化旨在用图形表现高维度的数据，并辅以交互手段，帮助人们分析和理解高维数据。

高维数据可视化主要分为降维方法和非降维方法。

1. 降维方法

降维方法将高维数据投影到低维空间，尽量保留高维空间中原有的特性和聚类关系。常见的降维方法有主成分分析（PCA）、多维度分析（Multi-Dimensional Scaling，MDS）和自组织图（Self-Organization Map，SOM）等。这些方法通过数学方法将高维数据降维，进而在低维屏幕空间中显示。通常，数据在高维空间中的距离越近，在投影图中两点的距离也越近。高维投影图可以很好地展示高维数据间的相似度以及聚类情况等，但并不能表示数据在每个维度上的信息，也不能表现维度间的关系。高维投影图损失了数据在原始维度上的细节信息，但直观地提供了数据之间宏观的结构。

常用的数据降维方法如图 3-9 所示。

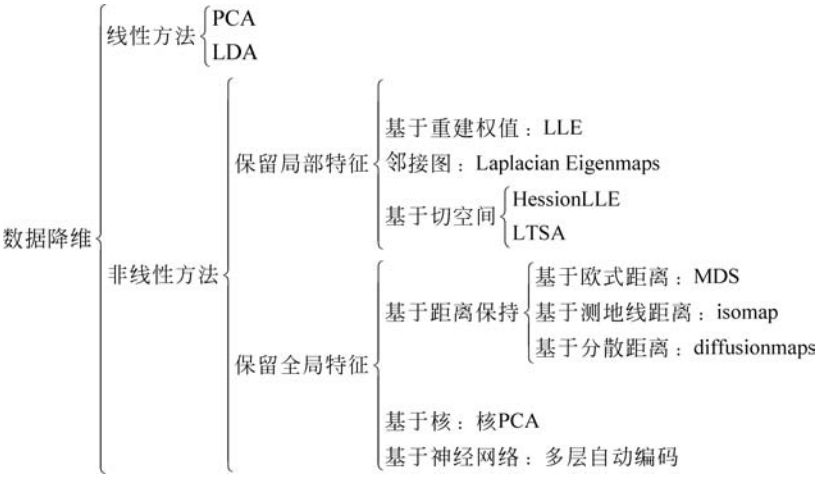

图 3-9　常用的数据降维方法

2. 非降维方法

非降维方法保留了高维数据在每个维度上的信息，可以展示所有的维度。各种非降维方法的主要区别在于如何对不同的维度进行数据到图像属性的映射。当维度较少时，可以直接通过与位置、颜色、形状等多种视觉属性相结合的方式对高维数据进行编码。当维度数量增多，数据量变大，或对数据呈现精度的需要提高时，这些方法难以满足需要。

3.3.7　Python 可视化

1. 简介

Python 在数据科学中的地位，不仅仅是因为 NumPy、SciPy、Pandas 和 scikit-learn 这些高效易用、接口统一的科学计算包，其强大的数据可视化工具也是重要组成部分。

在 Python 中，使用最多的数据可视化工具是 Matplotlib，除此之外，还有很多其他可选的可视化工具包，主要包括以下几类。

（1）Matplotlib 以及基于 Matplotlib 开发的工具包：Pandas 中的封装 Matplotlib API 的画图功能，Seaborn 和 networkx 等；

（2）基于 JavaScript 和 d3.js 开发的可视化工具，如 plotly 等，这类工具可以显示动态图且具有一定的交互性；

（3）其他提供了 Python 调用接口的可视化工具，如 OpenGL、GraphViz 等，这一类工具各有特点且在特定领域应用广泛。

对于数据科学，用的比较多的是 Matplotlib 和 Seaborn，对数据进行动态或交互式展示时会用到 plotly。

2. Python 数据可视化示例

【例 3-8】 Python 词云绘制示例。

```
In[103]:    from pyecharts import options as opts
            from pyecharts.charts import Page, WordCloud
            from pyecharts.globals import SymbolType
            words = [
                ("牛肉面", 7800),("黄河", 6181),
                ("«读者»杂志", 4386), ("甜胚子", 3055),
                ("甘肃省博物馆", 2055),("莫高窟", 8067),("兰州大学", 4244),
                ("西北师范大学", 1868),("中山桥", 3484),
                ("月牙泉", 1112),("五泉山", 980),
                ("五彩丹霞", 865),("黄河母亲", 847),("崆峒山",678),
                ("羊皮筏子", 1582),("兴隆山",868),
                ("兰州交通大学", 1555),("白塔山", 2550),("五泉山", 2550)]
            c = WordCloud()
            c.add("", words, word_size_range = [20, 80])
            c.set_global_opts(title_opts = opts.TitleOpts(title = "WordCloud - 基本示例"))
            c.render_notebook()
```

输出结果如图 3-10 所示。

图 3-10　词云

绘制回归图可以揭示两个变量间的线性关系。Seaborn 中使用 regplot()函数绘制回归图。

【例 3-9】 使用 regplot()函数绘制回归图。

```
In[104]:    import seaborn as sns
            df_iris = sns.load_dataset('iris')
            df_iris.head()
            # sns.barplot(x = df_iris['species'],y = df_iris['petal_length'],
                          data = df_iris)
            sns.regplot(x = 'petal_length',y = 'petal_width',data = df_iris)
```

输出结果如图 3-11 所示。

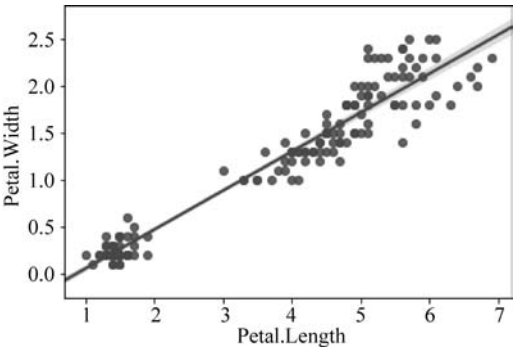

图 3-11 回归图


视频讲解

3.4 数据对象的相似性度量

现实中,我们需要处理的数据具有不同的形式和特征。而对数据相似性的度量又是数据挖掘分析中非常重要的环节。针对这些不同形式的数据,不可能找到一种具备普遍意义的相似性度量算法,甚至可以说,每种类型的数据都有它对应的相似度度量标准。

3.4.1 数据矩阵和相异性矩阵

数据矩阵(Data Matrix)又称为对象-属性结构,这种数据结构用关系表的形式或 $n \times p$(n 个对象,p 个属性)矩阵存放 n 个数据对象,每行对应一个对象。数据矩阵如下所示。

$$\begin{bmatrix} O_{11} & \cdots & O_{1f} & \cdots & O_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ O_{i1} & \cdots & O_{if} & \cdots & O_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ O_{n1} & \cdots & O_{nf} & \cdots & O_{np} \end{bmatrix}$$

相异性矩阵(Dissimilarity Matrix)也称为对象-对象结构,存放 n 个对象两两之间的邻近度,是 $n\times n$ 的矩阵,如下所示。

$$\begin{bmatrix} 0 & d(1,2) & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ \vdots & \vdots & \vdots & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

其中,$d(i,j)$为对象 i 和对象 j 之间的相异性的度量,且 $d(i,j)=d(j,i)$,因此相异性矩阵是对称矩阵。一般来说,$d(i,j)$是一个非负数。当 i 和 j 高度相似或"接近"时,它的值接近于 0;反之,差异越大时,这个值也越大。

相似性度量可以表示成相异性度量的函数,如式(3.6)所示。

$$\text{sim}(i,j)=1-d(i,j) \tag{3.6}$$

其中,$\text{sim}(i,j)$为对象 i 和对象 j 之间的相似性。

3.4.2 标称属性的相似性度量

两个对象 i 和 j 之间的相异性根据不匹配率进行计算,如式(3.7)所示。

$$d(i,j)=\frac{p-m}{p}=1-\frac{m}{p} \tag{3.7}$$

其中,m 为匹配的数目,即对象 i 和 j 状态相同的属性数;p 为对象的属性总数。

3.4.3 二元属性的相似性度量

二元属性只有两个状态,通常表示为 1(True)或 0(False)。在计算两个二元属性间的相异性时,涉及由给定的二元数据计算相异性矩阵。如果所有的二元都被看作具有相同的权重,可以得到一个两行两列的列联表,如表 3-2 所示。

表 3-2 二元属性的列联表

对象 i \ 对象 j	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

表 3-2 中,q 为对象 i 和 j 都取 1 的属性数;r 为对象 i 取 1,对象 j 取 0 的属性数;s 为对象 i 取 0,对象 j 取 1 的属性数;t 为对象 i 和 j 都取 0 的属性数。

对于对称的二元属性,两个状态是同等重要的。如果对象 i 和 j 都用对称的二元属性刻画,则 i 和 j 的相异性定义为

$$d(i,j)=\frac{r+s}{q+r+s+t} \tag{3.8}$$

对于非对称二元属性,只关心"正匹配"的情况,也就是只关心两个对象属性都取 1

的情况，因此负匹配数 t 被认为是不重要的，可以忽略，如式(3.9)所示。

$$d(i,j)=\frac{r+s}{q+r+s} \tag{3.9}$$

互补地，可以用基于相似性而不是相异性度量两个二元属性的差别。对象 i 和 j 之间的非对称的二元相似性可以表示为

$$\mathrm{sim}(i,j)=1-d(i,j)=\frac{q}{q+r+s} \tag{3.10}$$

如果把两个对象看作两个集合，相当于两个集合的交集比两个集合的并集。所以，式(3.10)不仅可以应用于二元属性，也可以应用于对两个集合相似度的度量，这个公式也叫作 Jaccard 系数。比较普遍的写法如式(3.11)所示。

$$\mathrm{sim}(U,V)=\frac{|U\cap V|}{|U\cup V|} \tag{3.11}$$

其中，U 和 V 代表两个集合，不一定具有相同数量的元素。

【例 3-10】 表 3-3 给出了居民家庭情况调查，包括属性姓名、婚姻状况、是否有房、是否有车 4 个属性。计算 3 名对象间的相异性。

表 3-3 居民家庭情况调查表

姓　名	婚姻状况	是否有房	是否有车
Zhang	Y	N	Y
Li	N	Y	Y
Wang	Y	Y	N

由数据可以看出，对象的二元属性是对称的，因此，根据式(3.8)得到 3 名对象间的相异性分别为

$$d(\mathrm{Zhang},\mathrm{Li})=\frac{1+1}{1+1+1}=0.67$$

$$d(\mathrm{Zhang},\mathrm{Wang})=\frac{1+1}{1+1+1}=0.67$$

$$d(\mathrm{Li},\mathrm{Wang})=\frac{1+1}{1+1+1}=0.67$$

3.4.4 数值属性的相似性度量

对于属性可以定量的属性类型，就叫作数值属性，关于这些属性值的分析可以说是最多的，常见的均值、众数、中位数等，就是处理这些属性的。数值属性的对象相似度一般用数据对象间的距离度量。

1. 欧氏距离

欧氏距离(Eulidean Distance)又称为直线距离。$i=(x_{i1},x_{i2},\cdots,x_{ip})$ 和 $j=(x_{j1},x_{j2},\cdots,x_{jp})$ 表示两个数值属性描述的对象。对象 i 和 j 之间的欧氏距离为

$$d(i,j)=\sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+\cdots+(x_{in}-x_{jp})^2} \tag{3.12}$$

2. 曼哈顿距离

曼哈顿距离(Manhattan Distance)又称为城市块距离,名称的由来是计量由方块形构成的曼哈顿街区的距离,因为街区不能横穿,只能按照方格走到。对象 i 和 j 之间的曼哈顿距离为

$$d(i,j)=|x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+\cdots+|x_{ip}-x_{jp}| \tag{3.13}$$

3. 切比雪夫距离

切比雪夫距离(Chebyshev Distance)是向量空间中的一种度量,两个数据对象 i 和 j 之间的切比雪夫距离定义为

$$d(i,j)=\lim_{k\to\infty}\left(\sum_{f=1}^{p}|x_{if}-x_{jf}|^k\right)^{1/k}=\max_{f\to p}|x_{if}-x_{jf}| \tag{3.14}$$

4. 闵可夫斯基距离

将曼哈顿距离与欧氏距离推广,可以得到闵可夫斯基距离(Minkowski Distance),如式(3.15)所示。

$$d(i,j)=\sqrt[p]{|x_{i1}-x_{j1}|^p+|x_{i2}-x_{j2}|^p+\cdots+|x_{ip}-x_{jp}|^p} \tag{3.15}$$

曼哈顿距离与欧氏距离是闵可夫斯基距离的两种特殊情形。

5. 汉明距离

两个等长字符串之间的汉明距离(Hamming Distance)定义为将其中一个变为另外一个所需要做的最小替换次数。例如,字符串"1111"与"1001"之间的汉明距离为 2。

3.4.5 序数属性的相似性度量

序数属性的每个属性值都代表了一种次序,所以,无论使用数字还是文字性的叙述,都可以表示成数字的形式。令序数属性可能的状态数为 M,这些有序的状态定义了一个排位 $1,2,\cdots,M_f$。

假设 f 是用于描述 n 个对象的一组序数属性之一,关于 f 的计算过程如下。

(1) 第 i 个对象的 f 值为 x_{if},属性 f 有 M_f 个有序的状态,表示排位 $1,2,\cdots,M_f$。用对应的排位 r_{if} 取代 x_{if}。

(2) 由于每个序数属性都可以有不同的状态数,所以通常将每个属性的取值映射到[0,1]上。通过用 z_{if} 代替第 i 个对象的 r_{if} 实现数据规格化,如式(3.16)所示。

$$z_{if}=\frac{r_{if}-1}{M_f-1} \tag{3.16}$$

(3) 相异性用任一种数值属性的距离度量,使用 z_{if} 作为第 i 个对象的 f 值。

3.4.6 混合类型属性的相似性

以上几种情况都是针对数据库中单一类型的数据，但是很多时候，遇到的一组数据可能拥有多种类型的属性，也就是混合类型属性。混合类型属性的相异性计算方法如式(3.17)所示。

$$d(i,j)=\frac{\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}}{\sum_{f=1}^{p}\delta_{ij}^{(f)}} \tag{3.17}$$

其中，$\delta_{ij}^{(f)}$ 为指示符，如果对象 i 或对象 j 没有属性 f 的度量值，或 $x_{if}=x_{jf}=0$ 时，$\delta_{ij}^{(f)}=0$，否则 $\delta_{ij}^{(f)}=1$。

3.4.7 余弦相似性

针对文档数据的相似度测量一般使用余弦相似性。在处理文档的时候，一般采用文档所拥有的关键词刻画一个文档的特征。容易想象，如果能定义一个字典(所谓字典，就是包含了所处理文档集中所有可能的关键词的一个有序集合)，那么就能通过字典为每个文档生成一个布尔型的向量，这个向量与字典等长，每个位置用 0/1 表示字典中对应的关键词在该文档的存在性。

这种方式与二元属性基本相同。但是，如果为了实现一种更准确的度量，需要给这个二元向量加个权重，如每个词的词频，这时使用之前提到的任何度量方法就都不太合适。例如，如果用欧氏距离判断相似度，因为这种向量很多位都是 0，即很稀疏，这就导致大部分词是两个文档所不共有的，从而判断结果是两个文档很不相似。对于这种文档-关键词的特殊情形一般采用余弦相似度，如式(3.18)所示。

$$\mathrm{sim}(\boldsymbol{x},\boldsymbol{y})=\frac{\boldsymbol{x}\cdot\boldsymbol{y}}{\|\boldsymbol{x}\|\cdot\|\boldsymbol{y}\|} \tag{3.18}$$

其中，$\boldsymbol{x}$ 和 $\boldsymbol{y}$ 是两个文档解析出来的词频向量。

假设有两个词频向量，$\boldsymbol{x}=\{3,0,4,2,0,6,2\}$，$\boldsymbol{y}=\{1,0,3,1,1,4,1\}$，则两个向量的余弦相似性计算为

$$\boldsymbol{x}\cdot\boldsymbol{y}=3\times1+0\times0+4\times3+2\times1+0\times1+6\times4+2\times1=43$$

$$\|\boldsymbol{x}\|=\sqrt{3^2+0^2+4^2+2^2+0^2+6^2+2^2}\approx8.31$$

$$\|\boldsymbol{y}\|=\sqrt{1^2+0^2+3^2+1^2+1^2+4^2+1^2}\approx5.39$$

$$\mathrm{sim}(\boldsymbol{x},\boldsymbol{y})=0.96$$

由此得到两个向量的余弦相似度为 0.96，说明两篇文档具有较高的相似性。

3.4.8 距离度量 Python 实现

【例 3-11】 设 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 为两个向量，长度都为 N，求它们之间的距离 d。

1. 用 NumPy 实现常见的距离度量

```
In[105]:    import numpy as np
            # 欧氏距离(Eulidean distance)
            def euclidean(x, y):
                return np.sqrt(np.sum((x - y) ** 2))
            # 曼哈顿距离(Manhattan distance)
            def manhattan(x,y):
                return np.sum(np.abs(x - y))
            #切比雪夫距离(Chebyshev distance)
            def chebyshev(x,y):
                return np.max(np.abs(x - y))
            # 闵可夫斯基距离(Minkowski distance)
            def minkowski(x,y,p):
                return np.sum(np.abs(x - y) ** p) ** (1/p)
            # 汉明距离(Hamming distance)
            def hamming(x, y):
            return np.sum(x!= y)/len(x)
            # 余弦距离
            def cos_similarity(x, y):
            return np.dot(x,y)/(np.linalg.norm(x) * np.linalg.norm(y))
```

2. 使用 SciPy 的 pdist ()方法进行数据对象的距离计算

代码格式为：scipy. spatial. distance. pdist(X，metric＝'euclidean'， * args， ** kwargs)。参数 X 为 m 个在 n 维空间上的观测值，参数 metric 为使用的距离度量，常用的取值有'canberra''chebyshev''cityblock''correlation''cosine''dice''euclidean''hamming''jaccard''jensenshannon''kulsinski''mahalanobis''matching'和'minkowski'等。

```
In[106]:    import numpy as np
            from scipy.spatial.distance import pdist
            x = (0.7,0.9,0.2,0.3,0.8,0.4,0.6,0,0.5)
            y = (0.6,0.8,0.5,0.4,0.3,0.5,0.7,0.2,0.6)
            X = np.vstack([x,y])
            d1 =  pdist(X, 'euclidean')
            print('欧氏距离: ',d1)
            d2 = pdist(X,'cityblock')
            print('曼哈顿距离: ',d2)
            d3 = pdist(X,'chebyshev')
            print('切比雪夫距离: ',d3)
            d4 = pdist(X,'minkowski',p = 2)
            print('闵可夫斯基距离: ',d4)
            d5 = pdist(X,'cosine')
            print('余弦相似性: ',1 - d5)
Out[106]:   欧氏距离: [0.66332496]
            曼哈顿距离: [1.6]
```

```
切比雪夫距离: [0.5]
闵可夫斯基距离: [0.66332496]
余弦相似性: [0.92032116]
```

3.5 小结

(1) 数据集由数据对象组成。数据对象代表实体,用属性描述。

(2) 数据属性有标称的、二元的、序数的或数值的。标称属性的值是一些符号或事物的名称,但可以用数字表示这些符号或名称,标称属性的值是枚举的;二元属性是标称属性的特例,也是一种布尔属性,对应 0 和 1 两个状态;序数属性的可能值之间存在有意义的序或秩评定,但是相继值之间的差是未知的;数值属性是可以度量的量,用整数或实数值表示,数值属性可以是区间标度的或比率标度的。

(3) 数据的基本统计描述为数据预处理提供了分析的基础。数据概括的基本统计量包括度量数据中心趋势的均值、加权均值、中位数和众数,以及度量数据散布的极差、分位数、四分位数、四分位数极差、方差和标准差等。

(4) 数据可视化技术主要有基于像素的、几何投影、基于图符的和层次化方法。

(5) 数据对象的相似性度量用于聚类、离群点分析等应用中。相似度度量基于相似性矩阵,对每种属性类型或其组合进行相似度计算。

习题 3

(1) 假设有数据属性取值(以递增序)为 5,9,13,15,16,17,19,21,22,22,25,26,26,29,30,32,39,52。分别计算该列数的均值、中位数、众数,并粗略估计第一四分位数和第三四分位数,绘制该数据的箱线图。

(2) 数据的可视化技术主要有哪些? 简述高维数据可视化方法。

(3) 计算数据对象 $x=(2,4,3,6,8,2)$ 和 $y=(1,4,2,7,5,3)$ 之间的欧氏距离、曼哈顿距离和闵可夫斯基距离,其中闵可夫斯基距离中的 p 取值为 3。

(4) 简述标称属性、非对称二元属性、数值属性和词频向量的相似度评价方法。