第 5 章

人工智能数学基础

本章导读

学习人工智能的知识首先要打好数学基础，数学基础知识蕴含着处理智能问题的基本思想与方法，也是理解复杂算法的必备要素。人工智能技术建立在数学模型之上，要了解人工智能，首先要掌握必备的数学基础知识。本章介绍在人工智能领域常用的数学知识，包括线性代数、概率论及最优化问题求解等。

学习目标

- 能使用线性代数解决常见数学问题并阐述其在人工智能中的应用。
- 能使用概率论解决常见数学问题并阐述其在人工智能中的应用。
- 能阐述最优化问题及其在人工智能中的应用。

各例题知识要点

例 5.1 矩阵乘法
例 5.2 求方程组的最小二乘解
例 5.3 计算矩阵的逆
例 5.4 计算分块矩阵的逆
例 5.5 计算行列式的值——第一种方法
例 5.6 计算行列式的值——第二种方法
例 5.7 求特征值和特征向量
例 5.8 将矩阵转化为对角形
例 5.9 奇异值分解
例 5.10 求离散随机变量的期望与方差
例 5.11 求连续随机变量的期望与方差
例 5.12 判断相关性

5.1 数学与人工智能

人工智能领域离不开数学，不论是机器学习还是深度机器学习，都会用到大量的数学知识，要想学好人工智能，首先要拥有一定的数学基础。

人工智能需要通过应用数学进行建模，然后进行数据采集、数据挖掘，最后通过适当的方法进行模型验证。将规律性事物总结为数学模型，不仅要还原所见的现象，还要基于现象进行仿真、模拟实验，并在这样的模型上进一步调控策略、定量评估，最终才能预测研

判、解决问题。现代人工智能的产生与提出是在学科交叉中涌现的，但它需要精准的数学刻画，故而需要应用数学知识的介入，从而开展更深入的研究。

5.2　线 性 代 数

线性代数是代数学的一个分支，主要处理线性问题。线性问题是指数学对象之间的关系是以一次形式来表达的。线性代数诞生于求解线性方程组，行列式、矩阵及向量是处理线性问题的有力工具。线性代数在人工智能领域的应用越来越多，神经网络中的所有参数都被存储在矩阵中，而线性代数使矩阵运算变得更加快捷简便。深度学习算法在GPU上训练模型时，GPU可以并行地以向量和矩阵运算。图像在计算中被表示为按序排列的二维像素数组。视频游戏使用庞大的矩阵来产生令人炫目的游戏体验。

5.2.1　矩阵的概念及矩阵运算

标量(Scalar)：一个数。例如 $x=3$。

向量(Vector)：一个有序排列的列数。例如 $\boldsymbol{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$。

矩阵(Matrix)：由 $m\times n$ 个数 $a_{ij}(i=1,2,\cdots,m;j=1,2,\cdots,n)$ 排成 m 行 n 列的数表，称为 m 行 n 列矩阵，简称为 $m\times n$ 矩阵，记作：

$$\boldsymbol{A}_{mn}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{5.1}$$

a_{ij} 表示第 i 行第 j 列元素，简记为 $A=(a_{ij})_{m\times n}$，有时候，为了指明所讨论的矩阵的级数，可以把 $m\times n$ 矩阵写成 $\boldsymbol{A}_{mn}$。另外，只有完全一样的矩阵才称为相等。

特殊地，行数与列数都等于 n 的矩阵称为 n 阶矩阵或 n 阶方阵。矩阵就是一个数表，且它是数学中一个极重要的应用广泛的工具。

5.2.2　张量

张量(Tensor)是深度学习中的一个重要概念，是TensorFlow、PyTorch等很多深度学习框架的重要组成部分。深度学习中的很多运算与模型优化过程都是基于Tensor完成的。

张量的定义：从代数角度讲它是向量的推广，可以看作是一个多维数组。目的是能够创造更高维度的矩阵、向量。零阶张量为一个标量；一阶张量为向量；二阶张量为矩阵。

下面举例说明张量：

从表5-1中可以看到，0阶标量只有大小，例如 $\boldsymbol{x}$ 为3；1阶张量即向量，有大小和方向，例如 $\boldsymbol{v}=[1,2,3]$；2阶张量即矩阵，类似数据表，例如矩阵 $\boldsymbol{m}$，每行均为一个列表，则

$\boldsymbol{m}$ 是二维列表，即列表中包含列表；3 阶张量，是数据立方体，例如 $\boldsymbol{t}$，其中[1]，[2]，[3]分别是向量，这三个向量组成一个矩阵，这个矩阵是 t 中的一个元素。

表 5-1　Python 示例

阶	数学含义	Python 示例
1	标量	$\boldsymbol{x}=3$
2	向量	$\boldsymbol{v}=[1,2,3]$
3	矩阵	$\boldsymbol{m}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$
4	3 阶张量	$\boldsymbol{t}=[[[1],[2],[3]],[[4],[5],[6]],[[7],[8],[9]]]$
5	n 阶张量	…

5.2.3　向量和矩阵的应用

在计算机视觉中，声音、图像、视频、标注信息等均被保存在向量或矩阵中，如一幅图像会以矩阵的形式进行存储。在进行图像识别时，常常会把图像矩阵转换为向量，然后对向量进行降维处理和识别，如图 5-1 所示，将灰度图像存储在矩阵 $\boldsymbol{A}_{mn}$ 中。也可以将灰度图像按列或行存储为 $\boldsymbol{A}_{n\times m,1}=[a_{11}a_{21}a_{31}\cdots a_{12}\cdots a_{mn}]^{\mathrm{T}}$ 或 $\boldsymbol{A}_{m\times n,1}=[a_{11}a_{12}a_{13}\cdots a_{21}\cdots a_{mn}]^{\mathrm{T}}$。彩色图像则可以存储在三个矩阵中，分别为对应的红、绿、蓝三原色矩阵。

$$\longleftrightarrow \quad A_{mn}=\begin{bmatrix}a_{11}&a_{12}&a_{13}&\cdots&a_{1n}\\a_{21}&a_{22}&a_{23}&\cdots&a_{2n}\\a_{31}&a_{32}&a_{33}&\cdots&a_{3n}\\\vdots&\vdots&\vdots&\ddots&\vdots\\a_{m1}&a_{m2}&a_{m3}&\cdots&a_{mn}\end{bmatrix}$$

图 5-1　灰度图像与二维矩阵对应

5.2.4　矩阵的运算

矩阵的运算是矩阵之间的一些最基本的关系，本节介绍矩阵的加法、乘法，矩阵与数的乘法及矩阵的转置。我们确定一个数域 P，以下所讨论的矩阵全是由数域 P 中的数组成的。

1. 加法

设：

$$\boldsymbol{A}_{sn}=\begin{bmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&\vdots&\ddots&\vdots\\a_{s1}&a_{s2}&\cdots&a_{sn}\end{bmatrix} \tag{5.2}$$

$$\boldsymbol{B}_{sn}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{s1} & b_{s2} & \cdots & b_{sn} \end{bmatrix} \tag{5.3}$$

是两个 $s\times n$ 矩阵，则矩阵

$$\boldsymbol{C}_{sn}=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1}+b_{s1} & a_{s2}+b_{s2} & \cdots & a_{sn}+b_{sn} \end{bmatrix} \tag{5.4}$$

称为 A 与 B 的和，记为：

$$\boldsymbol{C}=\boldsymbol{A}+\boldsymbol{B} \tag{5.5}$$

矩阵的加法就是矩阵对应元素相加，当然，相加的矩阵必须要有相同的行数和列数。由于矩阵的加法归结为它们的元素的加法，也就是数的加法，所以不难验证，它有：

结合律：$\boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})=(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C}$；

交换律：$\boldsymbol{A}+\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{A}$。

元素全为零的矩阵称为零矩阵，记为 $\boldsymbol{O}$，对所有的 $\boldsymbol{A}$，有 $\boldsymbol{A}+\boldsymbol{O}=\boldsymbol{A}$。

矩阵：

$$\begin{bmatrix} -a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & -a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{s1} & -a_{s2} & \cdots & -a_{sn} \end{bmatrix} \tag{5.6}$$

称为矩阵 $\boldsymbol{A}$ 的负矩阵，记为 $-\boldsymbol{A}$，显然有 $\boldsymbol{A}+(-\boldsymbol{A})=\boldsymbol{O}$。

矩阵的减法定义为：$\boldsymbol{A}-\boldsymbol{B}=\boldsymbol{A}+(-\boldsymbol{B})$。

根据矩阵加法的定义应用关于向量组秩的性质，很容易看出：秩$(\boldsymbol{A}+\boldsymbol{B})\leqslant$秩$(\boldsymbol{A})+$秩$(\boldsymbol{B})$。

2. 乘法

设 $\boldsymbol{A}_{sn}$ 为 $s\times n$ 矩阵，$\boldsymbol{B}_{nm}$ 为 $n\times m$ 矩阵，那么矩阵 $\boldsymbol{C}_{sm}$，其中元素：

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{in}b_{nj}=\sum_{k=1}^{n}a_{ik}b_{kj} \tag{5.7}$$

称为 $\boldsymbol{A}_{sn}$ 与 $\boldsymbol{B}_{nm}$ 的乘积，记为 $\boldsymbol{C}=\boldsymbol{AB}$。

由矩阵乘法的定义可以看出，矩阵 $\boldsymbol{A}_{sn}$ 与 $\boldsymbol{B}_{nm}$ 的乘积 $\boldsymbol{C}_{sm}$ 的第 i 行第 j 列的元素等于第一个矩阵 $\boldsymbol{A}_{sn}$ 的第 i 行与第二个矩阵 $\boldsymbol{B}_{nm}$ 的第 j 列的对应元素乘积之和，当然，在乘积的定义中，我们要求第二个矩阵的行数与第一个矩阵的列数相等。

【例 5.1】 设矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$，求 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的乘积。

$$\boldsymbol{A}=\begin{bmatrix} 1 & 0 & -1 & 2 \\ -1 & 1 & 3 & 0 \\ 0 & 5 & -1 & 4 \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} 0 & 3 & 4 \\ 1 & 2 & 1 \\ 3 & -1 & -1 \\ -1 & 2 & 1 \end{bmatrix} \tag{5.8}$$

解：

$$\boldsymbol{C}=\boldsymbol{AB}=\begin{bmatrix}1&0&-1&2\\-1&1&3&0\\0&5&-1&4\end{bmatrix}\begin{bmatrix}0&3&4\\1&2&1\\3&-1&-1\\-1&2&1\end{bmatrix}=\begin{bmatrix}-5&8&7\\10&-4&-6\\-2&19&10\end{bmatrix} \tag{5.9}$$

矩阵 $\boldsymbol{C}$ 中各个元素是根据式(5.7)得出的，例如，第二行第一列的元素 10 是矩阵 $\boldsymbol{A}$ 的第二行元素与矩阵 $\boldsymbol{B}$ 的第一列的对应元素乘积之和：

$$(-1)\times 0+1\times 1+3\times 3+0\times(-1)=10$$

矩阵的乘法可用于坐标变换，如在空间中作为一坐标系的转轴，设坐标系(x_1,y_1,z_1)到(x_2,y_2,z_2)的坐标变换矩阵为：

$$\boldsymbol{A}=\begin{bmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{bmatrix} \tag{5.10}$$

如果令：

$$X_1=\begin{bmatrix}x_1\\y_1\\z_1\end{bmatrix},\quad X_2=\begin{bmatrix}x_2\\y_2\\z_2\end{bmatrix} \tag{5.11}$$

那么坐标变换的公式可以写成：

$$X_1=AX_2 \tag{5.12}$$

如果再次作为坐标系的转轴，设由第二个坐标系(x_2,y_2,z_2)到第三个坐标系(x_3,y_3,z_3)的坐标变换公式为：

$$X_2=\boldsymbol{B}X_3 \tag{5.13}$$

其中：

$$\boldsymbol{B}=\begin{bmatrix}b_{11}&b_{12}&b_{13}\\b_{21}&b_{22}&b_{23}\\b_{31}&b_{32}&b_{33}\end{bmatrix},\quad X_3=\begin{bmatrix}x_3\\y_3\\z_3\end{bmatrix} \tag{5.14}$$

那么，由第一个坐标系到第三个坐标系的坐标变换矩阵为：

$$\boldsymbol{C}=\boldsymbol{AB} \tag{5.15}$$

矩阵的乘法适合结合律：

$$(\boldsymbol{AB})\boldsymbol{C}=\boldsymbol{A}(\boldsymbol{BC}) \tag{5.16}$$

但不适合交换律，一般来说：

$$\boldsymbol{AB}\neq\boldsymbol{BA} \tag{5.17}$$

矩阵的乘法和加法适合分配率，即：

$$\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C})=\boldsymbol{AB}+\boldsymbol{AC} \tag{5.18}$$

$$(\boldsymbol{B}+\boldsymbol{C})\boldsymbol{A}=\boldsymbol{BA}+\boldsymbol{CA} \tag{5.19}$$

应该指出的是，由于矩阵的乘法不适合交换律，所以以上两式为不同的规律。

还可以定义矩阵的方幂，设 $\boldsymbol{A}$ 是一个 $n\times n$ 矩阵，定义

$$\begin{cases}\boldsymbol{A}^1=\boldsymbol{A}\\ \boldsymbol{A}^{k+1}=\boldsymbol{A}^k\boldsymbol{A}\end{cases}\tag{5.20}$$

换句话说，$\boldsymbol{A}^k$ 就是 k 个 $\boldsymbol{A}$ 连乘。当然，方幂只能对行数与列数相等的矩阵来定义，由乘法的结合律不难证明

$$\boldsymbol{A}^k\boldsymbol{A}^l=\boldsymbol{A}^{k+l}\tag{5.21}$$

$$(\boldsymbol{A}^k)^l=\boldsymbol{A}^{kl}\tag{5.22}$$

这里 k 和 l 是任意正整数。因为矩阵的乘法不适合交换律，所以 $(\boldsymbol{AB})^k$ 与 $\boldsymbol{A}^k\boldsymbol{B}^k$ 一般不相等。

3. 数量乘法

矩阵：

$$\begin{bmatrix}ka_{11} & ka_{12} & \cdots & ka_{1n}\\ ka_{21} & ka_{22} & \cdots & ka_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ ka_{s1} & ka_{s2} & \cdots & ka_{sn}\end{bmatrix}$$

称为矩阵 $\boldsymbol{A}_{sn}$ 与数 k 的数量乘积，记为 $k\boldsymbol{A}$，换句话说，用数 k 乘矩阵就是把矩阵的每个元素都乘上 k。数量乘积适合以下规律：

$$(k+l)\boldsymbol{A}=k\boldsymbol{A}+l\boldsymbol{A}\tag{5.23}$$

$$k(\boldsymbol{A}+\boldsymbol{B})=k\boldsymbol{A}+k\boldsymbol{B}\tag{5.24}$$

$$k(l\boldsymbol{A})=(kl)\boldsymbol{A}\tag{5.25}$$

$$1\boldsymbol{A}=\boldsymbol{A}\tag{5.26}$$

$$k(\boldsymbol{AB})=(k\boldsymbol{A})\boldsymbol{B}=\boldsymbol{A}(k\boldsymbol{B})\tag{5.27}$$

矩阵：

$$\boldsymbol{kE}=\begin{bmatrix}k & 0 & \cdots & 0\\ 0 & k & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & k\end{bmatrix}\tag{5.28}$$

通常称为数量矩阵，如果 $\boldsymbol{A}$ 是一个 $n\times n$ 矩阵，那么有：

$$k\boldsymbol{A}=(k\boldsymbol{E})\boldsymbol{A}=\boldsymbol{A}(k\boldsymbol{E})\tag{5.29}$$

式(5.29)说明，数量矩阵与所有的 $n\times n$ 矩阵作乘法是可交换的。

4. 转置

把一矩阵 $\boldsymbol{A}$ 的行列互换，所得到的矩阵称为 $\boldsymbol{A}$ 的转置，记为 $\boldsymbol{A}'$ 或 $\boldsymbol{A}^{\mathrm{T}}$，确切的定义如下：

设：

$$\boldsymbol{A}=\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ a_{s1} & a_{s2} & \cdots & a_{sn}\end{bmatrix}\tag{5.30}$$

所谓 $\boldsymbol{A}$ 的转置是指矩阵

$$\boldsymbol{A}'=\begin{bmatrix} a_{11} & a_{21} & \cdots & a_{s1} \\ a_{12} & a_{22} & \cdots & a_{s2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{sn} \end{bmatrix} \tag{5.31}$$

显然，$s\times n$ 矩阵转置是 $n\times s$ 矩阵。矩阵的转置适合以下规律：

$$(\boldsymbol{A}')'=\boldsymbol{A} \tag{5.32}$$

$$(\boldsymbol{A}+\boldsymbol{B})'=\boldsymbol{A}'+\boldsymbol{B}' \tag{5.33}$$

$$(\boldsymbol{AB})'=\boldsymbol{B}'\boldsymbol{A}' \tag{5.34}$$

$$(k\boldsymbol{A})'=k\boldsymbol{A}' \tag{5.35}$$

5.2.5 矩阵的应用

矩阵在模式识别算法中可以存放原始数据、距离矩阵，在参与运算时可存放中间数据等，下面简要介绍一下。

1. 存放原始数据

(1) 灰度图像：用一个二维矩阵表示。

可以将一幅灰度图像存储在一个二维矩阵中，例如 $\boldsymbol{A}_{mn}$ 中存放一幅灰度图像，则矩阵中每一个元素 a_{ij} 代表一个灰度值。可以对 $\boldsymbol{A}_{mn}$ 进行处理，如边缘提取、余弦变换等，处理后将得到一个新的矩阵。

$$\boldsymbol{A}_{mn}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ddots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \tag{5.36}$$

(2) 深度图像：用一个二维矩阵表示。

深度图像的生成方法为将三维数据各点垂直投影到一个固定平面，由此每个点在平面上产生一个投影点，将各点到这个固定平面的距离转化为投影点的灰度值，最后使用线性插值将各点灰度值规整到二维网格中。图 5-2 第二行为第一行中四个三维人脸对应的深度图像。

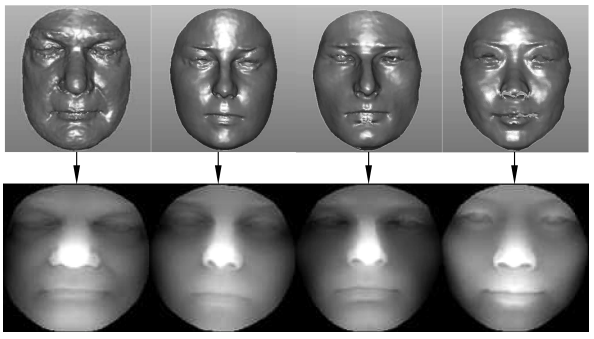
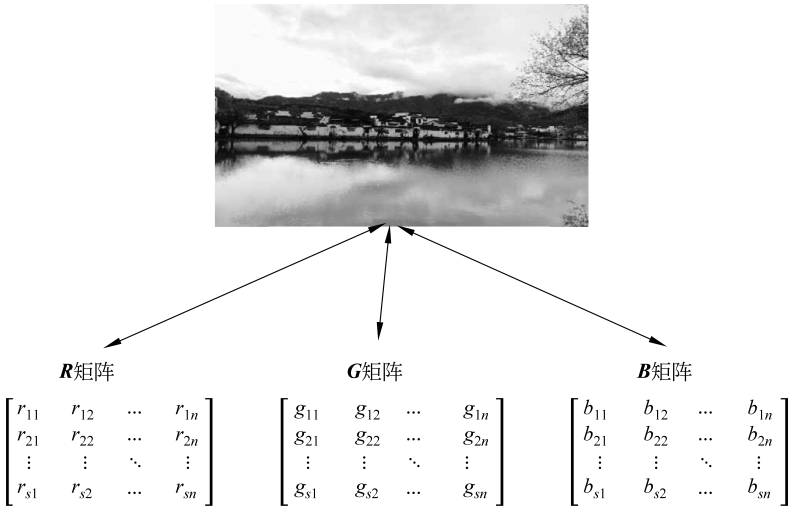


图 5-2　三维人脸深度图

(3) 彩色图像：用三个二维矩阵表示。

每一个像素点由 $\boldsymbol{R}$、$\boldsymbol{G}$、$\boldsymbol{B}$ 三个矩阵中对应的点表示。如图 5-3 中左上角第一个像素点由 $\boldsymbol{R}$、$\boldsymbol{G}$、$\boldsymbol{B}$ 三个矩阵中的(r_{11},g_{11},b_{11})表示。



$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{s1} & r_{s2} & \cdots & r_{sn} \end{bmatrix} \quad \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{s1} & g_{s2} & \cdots & g_{sn} \end{bmatrix} \quad \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{s1} & b_{s2} & \cdots & b_{sn} \end{bmatrix}$$

图 5-3　$\boldsymbol{R}$、$\boldsymbol{G}$、$\boldsymbol{B}$ 三个矩阵表示一幅彩色图像

(4) 三维数据：用四个二维矩阵表示。

该数据形式可将三维数据存储在四个大小相同的矩阵中，前三个矩阵($\boldsymbol{X}$ 矩阵、$\boldsymbol{Y}$ 矩阵、$\boldsymbol{Z}$ 矩阵)分别存储数据的三维坐标值，这三个矩阵中相同位置的点，例如[$\boldsymbol{X}(i,j)$，$\boldsymbol{Y}(i,j)$，$\boldsymbol{Z}(i,j)$]代表三维数据上一个点的空间位置坐标，如图 5-4 所示。第四个矩阵为有效点索引矩阵，白色部分索引值为 1，黑色部分索引值为 0。白色部分对应的前三个矩阵中相同位置的点为有效点，而黑色部分则对应无效点。

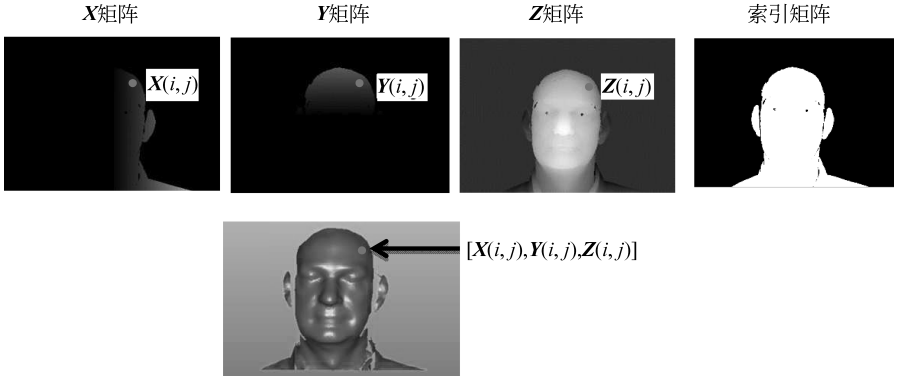


图 5-4　三维数据矩阵存储方式

这种存储格式有两个优点。其一可以表明各点的相对位置关系。其二这种数据形式可以变换为点云形式与深度图形式，即将矩阵中的有效点按顺序存储在 $N\times3$ 的矩阵中，可以得到点云数据，将 $\boldsymbol{Z}$ 矩阵中各点的值转化为深度值可得到数据的深度图。

2. 距离矩阵

例如在使用主成分分析算法(Principal Component Analysis,PCA)时,使用降维的数据集进行模式识别,将计算出的距离存储在距离矩阵中,通过分析距离矩阵中的数据,可以得出识别率。例如,下面的距离矩阵为测试集每个样本到注册集每个样本的距离组成的矩阵(本例的测试集与注册集中样本不同,而相同时的距离矩阵为对称矩阵),从矩阵中可分析出测试集中第1个样本与注册集中第2个样本的距离最短,因此将其判为注册集中第二个样本的所属类。

$$\begin{array}{c} \\ \text{注册集 gallery} \end{array}\begin{array}{c} \text{测试集 probe} \\ \begin{bmatrix} 0.2 & 0.4 & 0.63 & 0.73 & 0.3 & 0.4 & 0.5 \\ 0.1 & 0.07 & 0.21 & 0.43 & 0.73 & 0.34 & 0.23 \\ 0.23 & 0.6 & 0.3 & 0.43 & 0.35 & 0.92 & 0.32 \\ 0.15 & 0.34 & 0.2 & 0.34 & 0.35 & 0.3 & 0.34 \\ 0.78 & 0.45 & 0.43 & 0.34 & 0.24 & 0.2 & 0.76 \\ 0.53 & 0.34 & 0.32 & 0.43 & 0.39 & 0.2 & 0.75 \\ 0.43 & 0.54 & 0.44 & 0.36 & 0.39 & 0.1 & 0.4 \end{bmatrix} \end{array}$$

3. 参与计算

(1) 求线性方程组的解。解线性方程,实质上是利用矩阵简化计算过程。

(2) 解析几何中的坐标变换。如果只考虑坐标系的转轴(反时针方向转轴),那么平面直角坐标的变换公式为:

$$\begin{cases} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{cases} \tag{5.37}$$

其中θ为x轴与x'轴的夹角。显然,新旧坐标之间的关系完全可以通过公式中系数所排成的2×2矩阵表示出来:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

通常上面的矩阵称为坐标变换$\begin{cases} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{cases}$的矩阵。在空间的情形中,保持原点不动的仿射坐标系的变换有公式:

$$\begin{cases} x = a_{11}x' + a_{12}y' + a_{13}z' \\ y = a_{21}x' + a_{22}y' + a_{23}z' \\ z = a_{31}x' + a_{32}y' + a_{33}z' \end{cases} \tag{5.38}$$

同样,矩阵:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

称为式(5.38)变换矩阵。

(3) 规划问题。在讨论国民经济的数学问题中也常常用到矩阵。例如,假设在某一地区,某一种物资,比如说煤,有s个产地$A_1, A_2, \cdots, A_s$和n个销售地$B_1, B_2, \cdots, B_n$,那

么一个调运方案就可以用一个矩阵来表示：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \cdots & a_{sn} \end{bmatrix}$$

其中 a_{ij} 表示由产地 A_i 运到销售地 B_j 的数量。具体算法可参照运筹学中的相关算法。

(4) 最小二乘拟合。可以把统计的数据拟合成直线或者一些特定的曲线，其本质是高维空间到低维空间的投影。

【例 5.2】 求下列方程组的最小二乘解：

$$\begin{cases} 0.84x_1 + 0.56x_2 = 0.37 \\ 0.51x_1 + 0.36x_2 = 0.28 \\ 0.71x_1 + 0.53x_2 = 0.43 \\ 0.23x_1 + 0.75x_2 = 0.25 \end{cases} \tag{5.39}$$

解：令

$$\boldsymbol{A} = \begin{bmatrix} 0.84 & 0.56 \\ 0.51 & 0.36 \\ 0.71 & 0.53 \\ 0.23 & 0.75 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0.37 \\ 0.28 \\ 0.43 \\ 0.25 \end{bmatrix} \tag{5.40}$$

则原方程组可以表示为：

$$\boldsymbol{A}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{B} \tag{5.41}$$

最小二乘解所满足的线性方程组为：

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{B} \tag{5.42}$$

即

$$\begin{cases} 1.52x_1 + 1.20x_2 = 0.82 \\ 1.20x_1 + 1.29x_2 = 0.73 \end{cases} \tag{5.43}$$

解方程组得：

$$\begin{cases} x_1 = 0.35 \\ x_2 = 0.24 \end{cases} \tag{5.44}$$

(5) 信息压缩。例如传输视频时由于视频过大导致传输时间较长，因此可以对每一帧图像进行压缩，可以利用矩阵的各类分解公式或变换来进行压缩，如矩阵的余弦变换，收到压缩后的数据进行解压，就可以得到所传输的视频。在压缩时会丢失一小部分数据，但不影响视频播放效果。

(6) n 维向量也可以看成矩阵的特殊情形。n 维行向量就是 $1\times n$ 矩阵，n 维列向量就是 $n\times 1$ 矩阵。

4. 图像处理

我们以灰度图像为例，将图像中的每一个像素点的灰度值存储在矩阵中，然后对矩阵