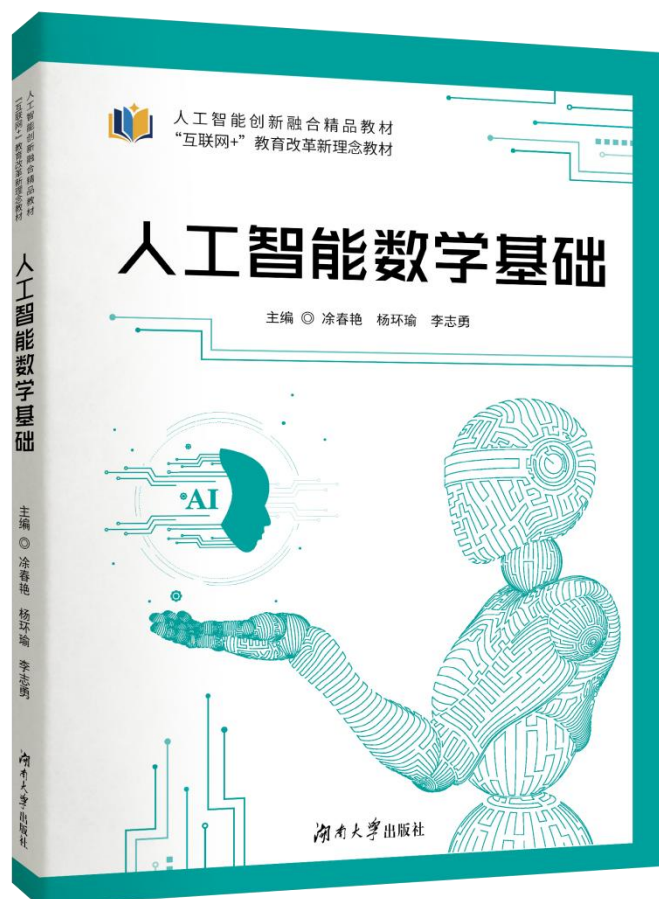


人工智能数学基础



类目：人工智能类

书名：人工智能数学基础

主编：涂春艳 杨环瑜 李志勇

出版社：湖南大学出版社

开本：大 16 开

书号：978-7-5667-5064-8

使用层次：通用

出版时间：2026 年 8 月

定价：59.80 元

印刷方式：双色

是否有资源：有



人工智能创新融合精品教材
“互联网+”教育改革新理念教材

人工智能数学基础

主编 ◎ 涂春艳 杨环瑜 李志勇



湖南大学出版社

人工智能创新融合精品教材
“互联网+”教育改革新理念教材

人工智能数学基础

主编 ◎ 涂春艳 杨环瑜 李志勇

湖南大学出版社



人工智能数学基础

责任编辑：杨琛晖
封面设计：旅语书装

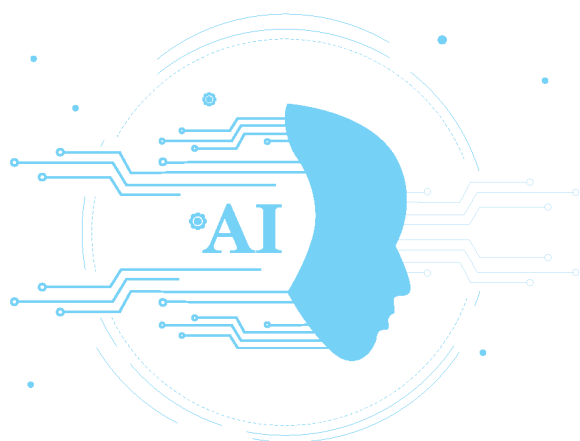




人工智能创新融合精品教材
“互联网+”教育改革新理念教材

人工智能数学基础

主 编 ◎ 涂春艳 杨环瑜 李志勇
副主编 ◎ 吕海侠 李德乐 毛晶晶
李海霞 张会丽 马旭东
叶 影 魏会贤 刘荷莲



湖南大学出版社

· 长 沙 ·

图书在版编目(CIP)数据

人工智能数学基础 / 涂春艳, 杨环瑜, 李志勇主编.
长沙: 湖南大学出版社, 2026. 8. -- ISBN 978-7-5667-5064-8

I. TP18; O29

中国国家版本馆 CIP 数据核字第 20261ZL367 号

人工智能数学基础

RENGONG ZHINENG SHUXUE JICHU

主 编: 涂春艳 杨环瑜 李志勇

责任编辑: 杨琛晖

印 装: 河北龙大印务有限公司

开 本: 889 mm×1194 mm 1/16 印 张: 16.5 字 数: 455 千字

版 次: 2026 年 8 月第 1 版 印 次: 2026 年 8 月第 1 次印刷

书 号: ISBN 978-7-5667-5064-8

定 价: 59.80 元

出 版 人: 李文邦

出版发行: 湖南大学出版社

社 址: 湖南·长沙·岳麓山 邮 编: 410082

电 话: 0731-88822559 (营销部) 88821174 (编辑室) 88821006 (出版部)

传 真: 0731-88822264 (总编室)

网 址: <http://press.hnu.edu.cn>

电子邮箱: xiaoshulianwenhua@163.com

版权所有, 盗版必究

图书凡有印装差错, 请与营销部联系



PREFACE

教育部于 2024 年启动了教育系统人工智能大模型示范应用行动，旨在打造人工智能通识课程体系，赋能理工医农文等各类人才培养，“人工智能+”时代来临，高校迫切需要构建一套完整的人工智能通识课程体系，为社会培养输送大量具有 AI 素养的优秀专业人才。

党的二十大报告提出要实施科教兴国战略，强化现代化建设人才支撑。强调要深化教育领域综合改革，加强教材建设和管理。为了响应党中央的号召，深入贯彻全国教育大会和《教育强国建设规划纲要（2024—2035 年）》精神，更好构建适应社会需求的人工智能通识课程教育体系，我们在充分调研和论证的基础上，精心编写了这本《人工智能数学基础》教材。

ChatGPT 一鸣惊人地将人工智能技术应用呈现在人们的面前，展示了人工智能技术颠覆性的力量，DeepSeek 的异军突起更是引起了整个社会关注人工智能的热潮。身处“人工智能+”时代，我们已经切身感受到了人工智能对人们的思维模式和生活方式的改变，人工智能技术必将对人类的社会生产和生活产生重大而深远的影响。与同类教材相比，本教材具备以下特点。

✓ 循序渐进，由浅入深

本教材系统性讲解生成式人工智能所涉及的相关数学理论，帮助读者形成对人工智能知识体系及其应用领域的理论性认识，为读者在人工智能领域的深造学习奠定基础。

✓ 说理透彻，注重实操

本教材在重视基本理论的前提下，还注重实际的可操作性，强调问题的解决，在每章均设有大量案例操作，以帮助读者巩固所学知识。

✓ 技能与思政教育紧密结合

本教材在讲解人工智能专业知识的同时，紧密结合思政教育主旋律，从专业知识角度触类旁通引导学生提升相关思政品质。

由于能力有限，书中难免存在不妥之处，恳请读者朋友提出宝贵意见，不胜感激。

编 者

2026 年 1 月



CONTENTS

第一部分 线性代数

第1章 向量空间与几何度量	2
1.1 向量与张量基础	3
1.1.1 标量、向量、矩阵与张量的定义	3
1.1.2 向量的线性运算	4
案例 1-1: 手机摄影美颜滤镜中的像素线性融合	6
1.1.3 线性无关与基变换	7
案例 1-2: 检测数据特征的冗余性	8
案例 1-3: 基变换实现数据表示的简化	10
1.2 范数理论与数据度量	11
1.2.1 范数与无穷范数	11
案例 1-4: 模型预测误差的风险评估	12
1.2.2 范数正则化	12
案例 1-5: L_1 与 L_2 正则化剔除成绩无效特征	13
1.2.3 距离度量	15
案例 1-6: 基于闵可夫斯基距离的用户偏好判定	15
1.3 内积空间与相似性	17
1.3.1 内积与几何投影	17
案例 1-7: 模拟卷积神经网络中的特征检测	18
1.3.2 相似性度量	19
案例 1-8: 相似度消除用户评分偏差的影响	19
1.3.3 正交向量组与标准正交基	20
案例 1-9: 标准正交基与特征的去相关性	21
第2章 矩阵分析与线性变换	23
2.1 矩阵运算与几何意义	24
2.1.1 矩阵乘法与线性映射视角	24
案例 2-1: 校园生活状态评估中的特征变换	24
2.1.2 逆矩阵与广义逆矩阵	25

案例 2-2: 利用广义逆矩阵求解薪资预测模型“无解”方程	26
2.1.3 行列式与几何体积缩放	27
案例 2-3: 图像变换中的面积缩放	30
2.2 特殊矩阵及其应用	31
2.2.1 对称矩阵与反对称矩阵	31
案例 2-4: 用户相似度矩阵的对称性验证	32
2.2.2 正交矩阵与旋转变换	33
案例 2-5: 数据增强中的坐标旋转变换	34
2.2.3 邻接矩阵与拉普拉斯矩阵	35
案例 2-6: 社交网络中的拓扑结构表征	38
2.3 矩阵秩与降维基础	40
2.3.1 秩的概念与数据线性相关性	40
案例 2-7: 传感器数据建模中的冗余检测	41
2.3.2 矩阵的低秩逼近与信息压缩	42
案例 2-8: 利用低秩逼近实现图像压缩	43
第3章 特征值与谱理论	45
3.1 特征值与特征向量	46
3.1.1 特征方程与特征空间的定义	46
案例 3-1: 验证线性变换中的特征不变方向	47
3.1.2 特征值的几何意义与伸缩属性	48
案例 3-2: 神经网络层对输入信号的增益控制	49
3.1.3 谱定理与矩阵的可对角化	49
案例 3-3: 通过对角化实现相关特征的解耦	50
3.2 二次型与正定性	51
3.2.1 二次型的矩阵表示	51
案例 3-4: 损失函数局部形态的矩阵表征	52
3.2.2 正定矩阵与半正定矩阵的判定	53
案例 3-5: 利用黑塞矩阵判定模型收敛状态	54
3.2.3 瑞利商与极值性质	56
案例 3-6: 脸特征降维任务中的最大投影方差搜索	57
第4章 矩阵分解与降维	59
4.1 奇异值分解 (SVD)	60
4.1.1 SVD 的定义与存在性	60
4.1.2 奇异值的能量分布意义	61
案例 4-1: 验证图像矩阵的能量集中现象	62
4.1.3 SVD 的低秩近似与图像压缩应用	65
案例 4-2: 边缘计算场景下的指纹图像压缩	65
4.2 主成分分析 (PCA)	68

4.2.1 协方差矩阵的特征分解	68
案例 4-3: 计算多特征数据的协方差矩阵	69
4.2.2 最大方差理论与主成分选取	70
案例 4-4: 基于累计贡献率确定降维维度	72
4.2.3 数据的白化处理	72
案例 4-5: 使用白化操作对特征进行规范化	74

第二部分 微积分与优化

第 5 章 多元微积分基础	77
----------------------------	-----------

5.1 导数与常用函数	78
5.1.1 导数的几何意义与物理意义	78
5.1.2 常见激活函数的导数	78
案例 5-1: 深度网络中的信号衰减数值分析	80
5.1.3 泰勒级数与局部线性近似	82
案例 5-2: 自动驾驶轨迹预测的线性化误差分析	83
5.2 偏导数与梯度	85
5.2.1 偏导数与全微分	85
5.2.2 梯度向量与方向导数	86
5.2.3 梯度的几何意义与最速下降方向	88
案例 5-3: 智能温控系统中的能耗最优化	90

第 6 章 矩阵微积分与自动微分	93
-------------------------------	-----------

6.1 雅可比矩阵与黑塞矩阵	94
6.1.1 向量对向量求导与雅可比矩阵	94
案例 6-1: 医疗生理特征映射中的雅可比灵敏度分析	95
6.1.2 二阶导数与黑塞矩阵	97
案例 6-2: 无人机悬停控制中的能量曲率分析	98
6.1.3 多元函数的极值判定与鞍点	100
案例 6-3: 黑塞矩阵特征值与函数几何形态的对比	102
6.2 矩阵求导法则	104
6.2.1 标量对向量/矩阵的求导	104
案例 6-4: 参数优化中的“病态”曲面下降	106
6.2.2 迹函数求导	108
案例 6-5: 神经网络数字图像特征图的重构	109
6.2.3 最小二乘法的矩阵求导	110
案例 6-6: 电商广告投入与销量的线性预测	112
6.3 计算图与反向传播	114
6.3.1 复合函数的链式法则	114

案例 6-7: 外卖预计送达时间 (ETA) 的敏感度分析	115
6.3.2 前向模式与反向模式微分	116
案例 6-8: 生成对抗样本欺骗图像识别系统	119
6.3.3 神经网络误差反向传播推导	120
案例 6-9: 基于误差反向传播算法的学情分析	122
第 7 章 凸优化理论	124
7.1 凸集与凸函数基础	125
7.1.1 仿射集与凸集的几何性质	125
案例 7-1: 图像像素颜色的凸组合变换	128
7.1.2 凸函数的一阶与二阶判定条件	129
案例 7-2: 情感分类任务中损失函数的局部凸性判定	130
7.1.3 Jensen 不等式及其应用	131
案例 7-3: 图像特征重构中的 KL 散度 (分布差异) 判定	133
7.2 凸优化问题的核心性质	134
7.2.1 凸优化问题的标准形式	134
案例 7-4: 算力调度任务中的线性规划标准形式构建	136
7.2.2 局部最优与全局最优性质	137
案例 7-5: 智能物流调度中的线性规划问题	139
7.2.3 典型凸优化问题分析	140
案例 7-6: 运动员精准膳食配比 (二次规划)	142
7.3 约束优化与对偶理论	144
7.3.1 拉格朗日乘子法: 从等式约束到不等式约束	144
7.3.2 对偶问题与弱强对偶性	145
7.3.3 KKT 条件与最优化求解	147
案例 7-7: 智能摄影中的噪点抑制与细节平衡应用	149
第 8 章 数值优化算法	151
8.1 一阶优化算法	152
8.1.1 梯度下降法 (GD) 的收敛性分析	152
案例 8-1: 智慧农业中的灌溉量优化	153
8.1.2 随机梯度下降 (SGD) 与小批量策略	155
案例 8-2: 电商推荐系统中的实时点击率 (CTR) 预估	156
8.1.3 动量法与 Nesterov 加速机制	158
案例 8-3: 自动驾驶汽车的精准限位停车	159
8.2 自适应学习率算法	161
8.2.1 基于梯度平方累积的 AdaGrad 与 RMSProp 算法	161
案例 8-4: 智能家居用电负荷的自适应优化	162
8.2.2 融合矩估计原理的 Adam 优化器及其数学推导	163
案例 8-5: 短视频推荐算法中的冷启动特征优化应用	165

第三部分 概率与信息

第9章 概率论基础	168
9.1 概率空间与事件	169
9.1.1 条件概率与全概率公式	169
案例 9-1: 智能医疗筛查中的患病置信度评估	170
9.1.2 独立性与相关性的区别	171
案例 9-2: 工业电机监测数据中的相关性与独立性分析	172
9.2 常用概率分布	174
9.2.1 伯努利分布与二项分布	174
案例 9-3: 社交媒体广告点击率 (CTR) 的统计推断	175
9.2.2 高斯分布 (正态分布) 与中心极限定理	177
案例 9-4: AI 辅助教学系统中的标准化考试成绩分析	178
9.2.3 多维高斯分布的几何性质	179
案例 9-5: 医疗影像分析中的病灶特征关联建模	181
9.3 随机变量的数字特征	183
9.3.1 期望、方差与协方差	183
案例 9-6: 视频直播流媒体传输的延迟波动评估	183
9.3.2 协方差矩阵与马氏距离的关系	184
案例 9-7: 生物制药实验中的基因表达协同分析	185
第10章 统计推断与参数估计	187
10.1 贝叶斯推断	188
10.1.1 先验概率、似然函数与后验概率	188
案例 10-1: 生物制药临床试验的参数收敛性分析	189
10.1.2 贝叶斯公式在机器学习中的视角	191
案例 10-2: 自动驾驶高噪声环境下的状态正则化	192
10.1.3 朴素贝叶斯分类原理	194
案例 10-3: 生产线轴承缺陷检测的概率判别实现	195
10.2 参数估计方法	197
10.2.1 极大似然估计 (MLE)	197
案例 10-4: 新能源电网风速概率特征的极大似然估计	198
10.2.2 最大后验估计 (MAP)	200
案例 10-5: 投资资产收益波动率的 MAP 估计	201
10.2.3 MLE 与最小二乘法及交叉熵的等价性	203
案例 10-6: 工业压力传感器校准的极大似然路径实现	204
第11章 信息论基础	207
11.1 信息的度量	208

11.1.1 自信息与香农熵	208
案例 11-1: 医疗影像诊断系统中的类别不确定性度量	209
11.1.2 联合熵与条件熵	211
案例 11-2: 新能源光伏发电预测中的不确定性量化应用	212
11.1.3 互信息与特征选择	215
案例 11-3: 肺部结节特征提取中的互信息特征选择应用	216
11.2 概率分布的距离	218
11.2.1 KL 散度 (相对熵)	218
案例 11-4: 智慧物流库存需求分布的预测偏差量化	219
11.2.2 交叉熵与 Loss 函数	221
案例 11-5: 智能制造中零件多分类任务的交叉熵损失量化	222
11.2.3 JS 散度与生成对抗网络 (GAN)	224
案例 11-6: 教育辅助系统中手写练习题合成质量的 JS 散度评估	226

第四部分 随机过程与决策

第 12 章 随机过程与强化学习基础	230
12.1 离散时间马尔可夫链	231
12.1.1 状态转移矩阵与状态图	231
案例 12-1: 新能源电力系统负荷状态转移的概率分析	232
12.1.2 查普曼-科尔莫戈罗夫方程 (C-K 方程)	234
案例 12-2: 电动汽车电池健康状态退化趋势评估	235
12.2 马尔可夫决策过程	237
12.2.1 状态、动作、奖励与策略	238
案例 12-3: 自适应教育系统中的教学策略反馈评估	239
12.2.2 回报与折扣因子	241
案例 12-4: 智慧物流路径规划中的长期回报评估应用	242
12.3 价值函数与贝尔曼方程	244
12.3.1 状态价值函数与动作价值函数	244
案例 12-5: 短视频留存优化中的动作价值评估应用	246
12.3.2 贝尔曼期望方程的递归形式	247
案例 12-6: 视频观看体验 (QoE) 优化中的递归价值评估	248
12.3.3 贝尔曼最优方程与动态规划思想	250
案例 12-7: 配送路径时效优化中的最优价值场计算	251
参考文献	254

第一部分 线性代数

第 1 章

向量空间与几何度量

本章导读

在人工智能与机器学习的实践中，所有数据（包括特征、参数、嵌入表示等）皆被抽象为向量和张量，它们构成了算法运行的抽象数据空间。

本章从数据的核心载体——标量、向量、矩阵与张量的定义着手，系统性探讨它们的线性运算及其在机器学习中的广播机制应用。同时，深入剖析线性无关、基变换等关键概念，为理解特征空间变换提供数学依据。接着，引入范数理论，用以量化数据的“规模”与复杂性，并结合距离度量评估数据点间的差异，这对理解损失函数设计和正则化技术（如 L_1 和 L_2 正则化）至关重要。最后，通过内积空间的概念，使学习者能够掌握如何度量数据间的相似性（例如余弦相似度），并学习利用正交化的方法构造高效、不相关的特征基。

本章的学习目标是使学习者掌握将数据对象几何化、在空间中进行操作与测量的基本工具，为深入理解后续的机器学习模型、神经网络结构及其优化理论奠定关键的几何思维基础。

学习指导

通过本章学习，你将：

- 理解与掌握标量、向量、矩阵与张量的定义，以及它们在 AI 数据表示中的核心作用。
- 熟练运用向量的线性运算，理解广播机制在张量操作中的效率优势。
- 掌握线性无关、基变换的几何意义，并理解其在特征空间重构中的应用。
- 能够计算范数和常见的距离度量（如欧氏距离），并解释范数在正则化中的作用。
- 理解内积的几何投影意义，并运用余弦相似度、皮尔逊相关系数来度量数据间的相似性。
- 理解正交向量组的概念，并掌握施密特正交化过程，为后续学习主成分分析（PCA）和特征分解打下基础。

1.1 向量与张量基础

在人工智能的数学建模过程中，客观世界的物理属性与逻辑特征需通过量化处理转化为特定的数学对象。如图 1-1 所示，不同复杂程度的数据特征对应着不同的代数结构：单一维度的物理量(如温度)被抽象为标量；具有大小与方向特征的有序数值组(如三维空间中的矢量)被抽象为向量；在二维平面上排列的像素阵列被抽象为矩阵；而多维度的复杂数据流(如时空耦合的视频序列)则被抽象为张量。这些代数对象不仅是存储数据的载体，更是执行线性变换与特征提取的数学基础。

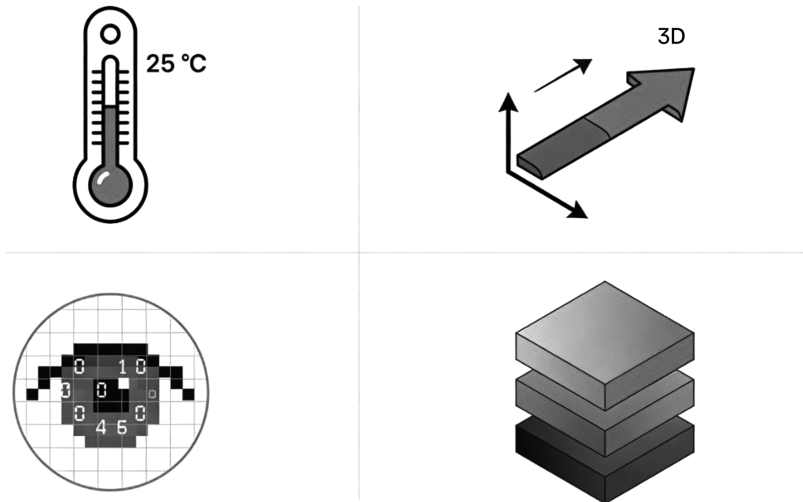


图 1-1 数据量化概念图

1.1.1 标量、向量、矩阵与张量的定义

在人工智能与机器学习中，数据是算法运算的核心对象。根据数据的维度(或阶数)不同，将数据容器分为四类：标量、向量、矩阵和张量。从数学本质上讲，它们构成了线性代数的基础层级结构。

(1) 标量

标量是一个独立的数值，它仅具有大小，没有方向。在几何意义上，标量代表零阶张量。通常用于表示损失函数的值、学习率、权重衰减系数等单一变量。通常使用斜体小写字母来表示标量，如 x 、 y 或 α 。一个实数标量 x 表示为 $x \in \mathbf{R}$ 。其中， $\mathbf{R}$ 表示实数集。例如，在智能传感器监测任务中，某一时刻采集到的环境温度即为一个标量。

(2) 向量

向量是由一组有序排列的标量组成的数组。它不仅具有大小，还具有方向。在几何意义上，向量代表一阶张量。在计算机科学中，它对应于一维数组。通常使用粗体斜体小写字母来表示向量，如 $\mathbf{x}$ 、 $\mathbf{y}$ 。向量中的元素(分量)则使用斜体非粗体小写字母表示，并加上下标，如 x_i 。

一个 n 维向量 $\mathbf{x}$ 包含 n 个实数元素。在数学推导中，默认向量为列向量。其标准定义如下：

$$\mathbf{x} \in \mathbf{R}^n$$

展开形式为：

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

有时为了排版方便，也可将其写为转置形式：

$$\mathbf{x} = [x_1, x_2, \dots, x_n]^T$$

其中， x_i 表示向量 $\mathbf{x}$ 中的第 i 个元素。在自动驾驶的运动规划任务中，车辆的速度向量包含了横向与纵向的分量，表达了位移变化的动态特征。

(3) 矩阵

矩阵是一个二维的数值数组，由行和列组成。在几何意义上，矩阵代表二阶张量。矩阵常用于表示数据集（如行表示样本，列表示特征）或线性变换（如神经网络中的权重参数）。通常使用粗体斜体大写字母来表示矩阵，如 $\mathbf{A}$ 、 $\mathbf{W}$ 。

一个 m 行 n 列的实数矩阵 $\mathbf{A}$ 定义为：

$$\mathbf{A} \in \mathbf{R}^{m \times n}$$

展开形式为：

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

其中，元素 a_{ij} （或写作 $A_{i,j}$ ）表示矩阵 $\mathbf{A}$ 中第 i 行、第 j 列的数值。在医疗影像处理中，一张黑白超声图像被量化为一个像素强度矩阵。

(4) 张量

张量是标量、向量和矩阵向任意维度的推广，它是存储数据的通用容器。

- 零阶张量是标量。
- 一阶张量是向量。
- 二阶张量是矩阵。
- 三阶及以上张量统称为高阶张量（如 RGB 图像数据通常表示为三阶张量：高度×宽度×通道数）。

通常使用手写体大写字母或特殊的无衬线字体来表示高阶张量，如 $\mathcal{X}$ 、 $\mathcal{Y}$ 。

一个 d 阶张量 $\mathcal{X}$ ，其各维度的长度分别为 $n_1, n_2, \dots, n_d$ ，定义为：

$$\mathcal{X} \in \mathbf{R}^{n_1 \times n_2 \times \cdots \times n_d}$$

张量中的元素由 d 个索引确定，记为 $x_{i_1, i_2, \dots, i_d}$ ，其中该元素位于第 1 维的 i_1 位置、第 2 维的 i_2 位置……依此类推。

1.1.2 向量的线性运算

向量空间的代数结构由向量加法与标量乘法两种基本运算定义。在数字图像处理领域，这两种运算构成了图像特征表示、信号强度增强以及多模态数据融合的数学基础。通过对像素向量执行线性组合，可以实现从底层数值计算到高层特征变换的逻辑演进。

(1) 向量的基本线性运算

向量的线性运算是构建高维空间特征表示的基础。通过对离散数据点执行代数加法与尺度缩放,可以实现特征维度的逻辑组合。

① 向量加法

设有两个同维度的 n 维向量 $\mathbf{x}, \mathbf{y} \in \mathbf{R}^n$, 其加法运算定义为对应分量的代数相加。设生成的和向量为 $\mathbf{z} = \mathbf{x} + \mathbf{y}$, 其分量表达式为:

$$z_i = x_i + y_i, i = 1, 2, \dots, n$$

由几何性质可知, 向量加法满足平行四边形法则, 反映了特征空间中位移矢量的合成, 如图 1-2 所示。相当于从原点出发, 先进行 $\mathbf{x}$ 的位移, 接着进行 $\mathbf{y}$ 的位移, 最终到达的位置即为 $\mathbf{x} + \mathbf{y}$ 。在图像降噪任务中, 将同一场景的多帧采样向量进行加法运算, 其物理意义在于通过特征叠加提升信号强度, 从而抑制随机分布的噪声分量。

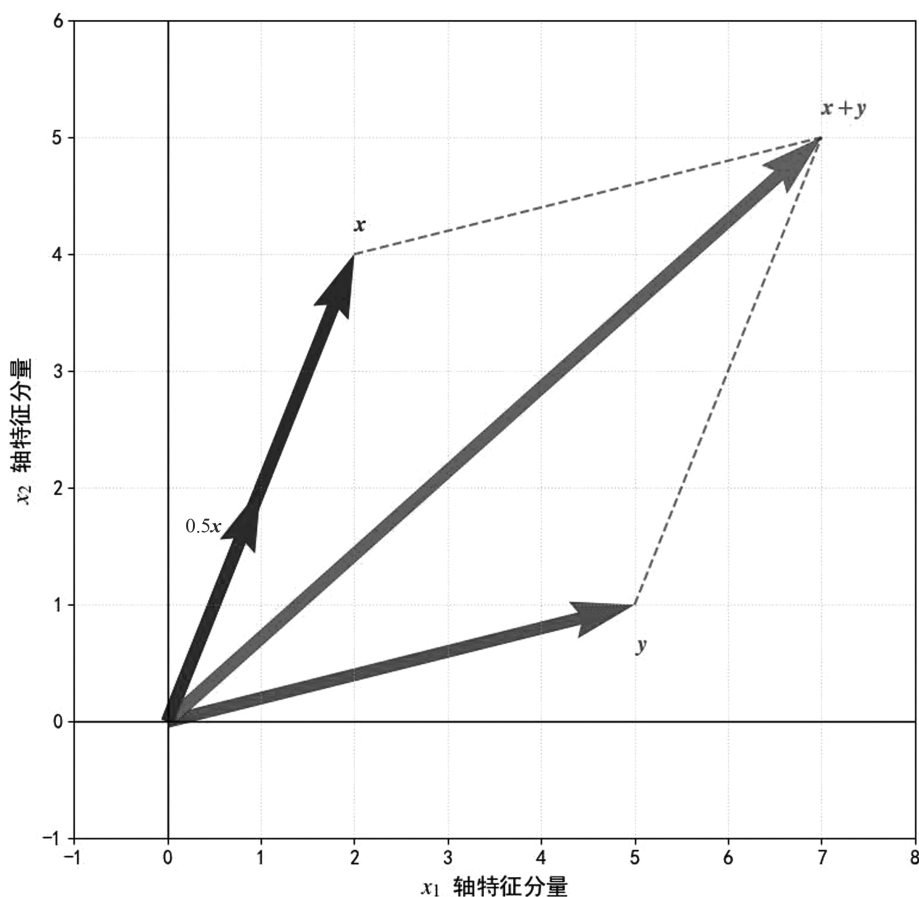


图 1-2 特征空间中向量加法与标量乘法的几何关系

② 标量乘法

设有一个标量 $\alpha \in \mathbf{R}$ 和一个向量 $\mathbf{x} \in \mathbf{R}^n$, 数乘定义为标量与向量的每一个分量相乘。设结果向量为 $\mathbf{z} = \alpha \mathbf{x}$, 则有:

$$z_i = \alpha x_i, i = 1, 2, \dots, n$$

标量乘法描述了特征向量模长的缩放现象。

- 若 $|\alpha| > 1$, 表现为特征强度在原方向上的线性拉伸。
- 若 $0 < |\alpha| < 1$, 表现为特征强度在原方向上的线性压缩, 如图 1-2 中的 $0.5\mathbf{x}$ 。

- 若 $\alpha < 0$ ，向量的方向发生反转。

在图像亮度调节任务中，标量乘法直接对应于增益系数的调整。

(2) 机器学习计算中的广播机制

在传统的数学定义中，两个数据进行加减运算时，其维度必须完全一致（例如矩阵加矩阵，行列数必须相同）。

然而，在人工智能和机器学习模型中，经常需要计算 $Y = WX + b$ ，其中 WX 可能是一个巨大的矩阵（包含很多样本），而偏置 b 可能只是一个短向量甚至一个标量。

为了简化代码编写，现代计算框架（如 NumPy、PyTorch、TensorFlow）引入了广播机制。它是一种允许不同形状的数组进行数学运算的算法规则。

当对两个形状不一致的数组执行线性运算时，系统遵循以下算法准则进行维度扩展：

- 对齐：将两个数组的维度从右向左对齐。
- 扩展：如果某个维度上两个数组的大小不一致，且其中一个大小为 1，则该维度会被“复制”或“延展”，直到与另一个数组的大小相同。
- 报错：如果两个数组在某个维度上的大小既不相等，其中一个也不是 1，则无法广播，抛出错误。

案例 1-1：手机摄影美颜滤镜中的像素线性融合

在手机摄影的滤镜算法中，最终输出的图像通常由原始场景像素向量与预设风格模板向量线性加权合成。本案例旨在利用向量的线性运算原理，通过调整权重系数 α 与 β ，实现图像亮度的特征融合。这一过程建立了从数学公式 $y = \alpha a + \beta b$ 到像素级图像增强任务的直接映射。

```
import pandas as pd
import matplotlib.pyplot as plt
# --- 全局字体统一配置 ---
plt.rcParams['font.sans-serif'] = ['SimHei'] # 中文建议使用黑体
plt.rcParams['axes.unicode_minus'] = False # 正常显示负号
# 1. 读取原始特征数据
df = pd.read_csv('pixel_data.csv')
a,b = df['raw'].values,df['style'].values

# 线性运算配置:30% 原始保留 + 70% 滤镜增强
alpha,beta = 0.3,0.7
fused_pixels = alpha * a + beta * b

# 结果可视化
plt.figure(figsize=(9,5))
plt.plot(a,'o--',label=r'原始亮度向量  $\mathbf{a}$ ')
plt.plot(b,'s--',label=r'风格模板向量  $\mathbf{b}$ ')
plt.plot(fused_pixels,'D-',label=r'融合结果向量  $\mathbf{y}$ ',linewidth=2)
plt.title('线性组合运算在像素级滤镜融合中的应用结果',fontsize=14)
plt.legend()
plt.grid(axis='y',linestyle=':',alpha=0.6)
plt.show()

print(f"融合后的像素特征向量为:{fused_pixels}")
```

运行结果：

融合后的像素特征向量为:[131. 151. 112.5 184. 222.5]

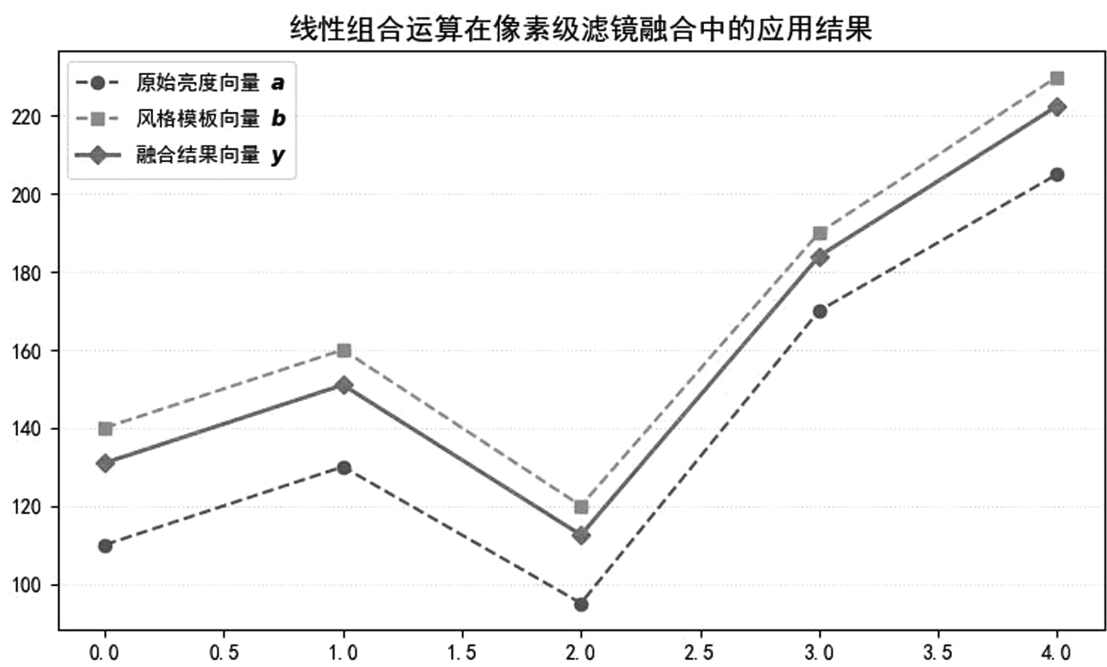


图 1-3 线性组合运算结果图

计算所得的融合向量分量严格遵循线性组合定义的代数映射。由图 1-3 可知，融合后的像素强度介于原始信号与模板信号之间。这证明了通过调节线性运算的标量权重，可以精准控制多源特征的信息占比，构成了机器学习中注意力权重分配与残差连接网络的最底层数学逻辑。

1.1.3 线性无关与基变换

在人工智能的数据处理中，经常面临两个核心问题：

- 特征是否冗余？（例如：“千米数”和“米数”是重复特征，需要剔除）。这对应数学上的线性相关性。
- 视角是否最佳？（例如：将数据旋转到主成分方向，以便降维）。这对应数学上的基变换（如图 1-4 所示）。

为了解决这些问题，需要研究向量内部的依赖关系，并由此确立空间的基准。

(1) 线性相关性与冗余特征判定

在探讨一组向量的性质时，首先需要判断它们之间是否存在线性依赖关系。在多维特征空间中，如果某一特征可以由其他特征的线性组合表达，则称该特征具有冗余性。在数学上，这一性质通过线性相关性进行描述。

给定向量空间 V 中的一组向量 $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ 和一组标量系数 $c_1, c_2, \dots, c_k$ ，形式如下的向量 $\mathbf{y}$ 称为这些向量的一个线性组合：

$$\mathbf{y} = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_k \mathbf{v}_k$$

也可以表示为：

$$\mathbf{y} = \sum_{i=1}^k c_i \mathbf{v}_i$$

若关于系数 c_i 的等式

$$\sum_{i=1}^k c_i \mathbf{v}_i = \mathbf{0}$$

仅在 $c_1 = c_2 = \dots = c_k = 0$ 时成立，则称该向量组线性无关；否则称其线性相关。由几何性质可知，线性无关的向量组中每一个向量都提供了独特的特征维度，无法被其他向量线性表示。

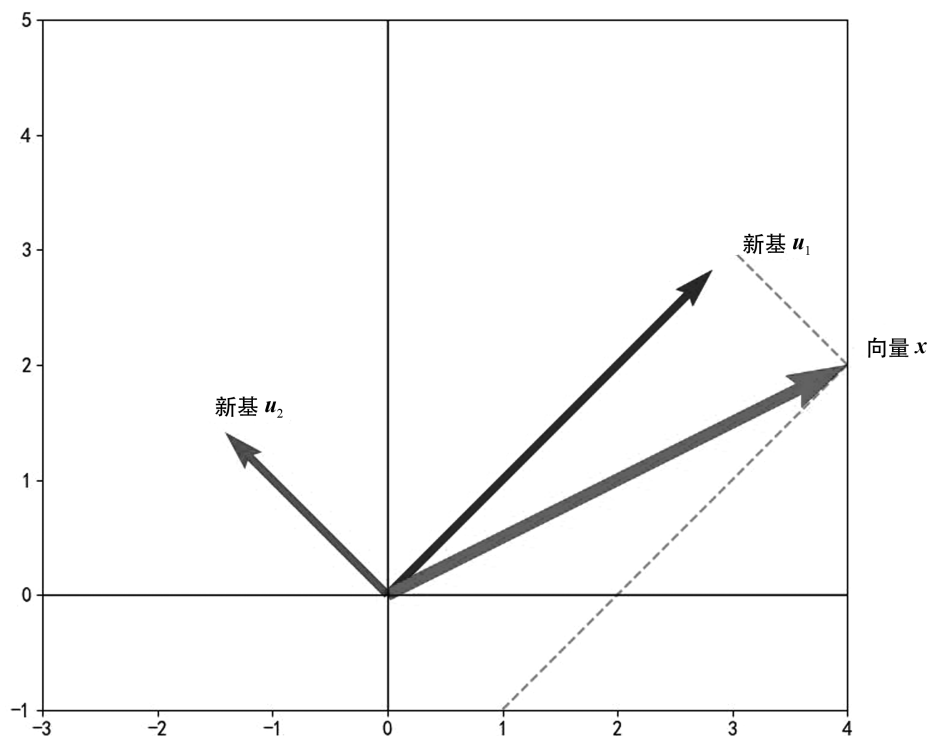


图 1-4 基变换与坐标表示

案例 1-2：检测数据特征的冗余性

在处理房屋数据时，提取了以下两个特征向量：

- 特征 A：房屋面积(单位：平方米) $\mathbf{v}_1 = [1, 2]$ 。
- 特征 B：房屋面积(单位：平方分米) $\mathbf{v}_2 = [100, 200]$ 。

显然，特征 B 只是特征 A 的 100 倍，存在线性相关。本案例演示如何通过 Python 代码发现这种依赖关系。

```
import numpy as np
# 定义特征向量
v1 = np.array([1, 2])    # 特征 A
v2 = np.array([100, 200]) # 特征 B(显然是 v1 的 100 倍)
# 寻找依赖关系(判断 v2 是否能被 v1 线性表示)
```

```
ratio = v2 / v1
print(f"特征 v1:{v1}")
print(f"特征 v2:{v2}")
print(f"倍数关系 (v2/v1):{ratio}")
```

运行结果:

```
特征 v1:[1 2]
特征 v2:[100 200]
倍数关系 (v2/v1):[100. 100.]
```

上面的结果证实了线性相关： v_2 是冗余特征，特征 B 可以剔除。

(2) 基与维数

在线性无关性的基础上，可以为向量空间确定一组最小的生成集，即基(basis)。这组基是向量空间的一种基本结构，空间中的任何元素都可以通过这组基来唯一表示。

①基的定义

向量空间 V 的一组向量 $B = \{e_1, e_2, \dots, e_n\}$ 若满足以下两个条件，则称为该空间的一组基：

- 线性无关： B 中的向量之间不存在线性依赖关系(即没有冗余向量)。
- 张成空间： V 中的任意向量都可以表示为 B 的线性组合。

②维数

向量空间 V 的基中所含向量的个数 n 是唯一的，称为该空间的维数，记为 $\dim(V) = n$ 。在人工智能任务中，维数 n 代表了特征空间的固有复杂度。

(3) 基变换

在不同的坐标系下，同一个客观存在的几何向量具有不同的数值表示。基变换是实现特征投影与维度压缩的核心数学工具。

设 n 维空间中有两组基：旧基(通常为标准基) $E = [e_1, \dots, e_n] = I$ (单位矩阵) 和新基 $U = [u_1, \dots, u_n]$ 。空间中的任意向量 x ，它在几何上是客观存在的，不随坐标系变化而变化。

设其在旧基下的坐标为 c_{old} ，在新基下的坐标为 c_{new} 。根据基的定义，向量 x 可以由基向量线性组合而成：

$$x = Ec_{old} = Uc_{new}$$

图 1-5 展示了二维空间中向量 x 在两组基下的坐标表示。

由于旧基 E 是单位矩阵($E=I$)，因此 $x=c_{old}$ 。此时，将新基向量按列排布构成的矩阵称为过渡矩阵 P (即 $P=U$)。于是有：

$$c_{old} = Pc_{new}$$

这个公式表明：新坐标左乘过渡矩阵，还原回旧坐标。在实际的特征提取应用中，通常已知旧坐标 c_{old} (原始数据)，要求其在新基下的新坐标 c_{new} (特征表示)。此时需对上式两边左乘 P^{-1} ：

$$c_{new} = P^{-1}c_{old}$$

若新基 U 是正交基(如在 PCA 或 SVD 中)，则 P 为正交矩阵，其逆矩阵等于转置矩阵，即 $P^{-1} = P^T$ ，这将极大地简化计算。

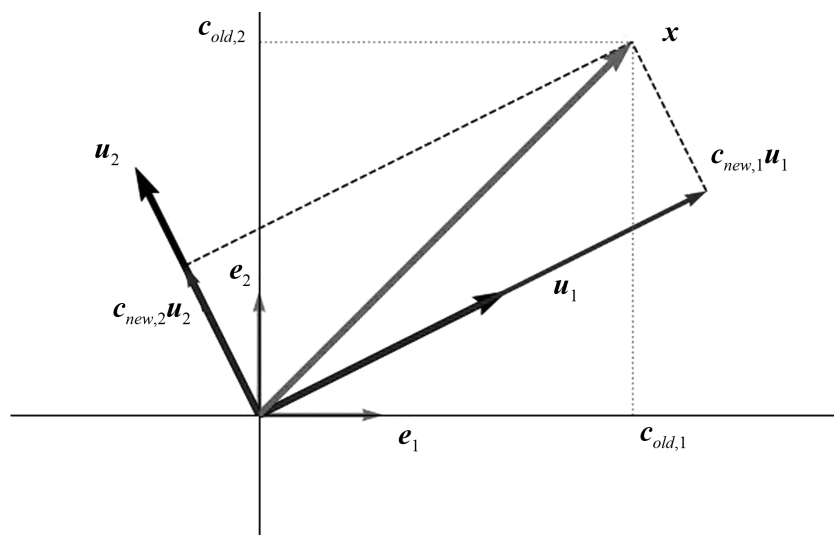


图 1-5 向量的基变换示意

案例 1-3：基变换实现数据表示的简化

在机器学习的特征提取任务中，原始数据往往存在冗余，导致维度过高且难以处理。基变换的核心作用是旋转“观察视角”，寻找一组能最大程度保留信息且去除冗余的新基。本案例模拟这一过程：将一个具有强相关性的数据点投影到一组顺应其分布规律的新基上，验证如何通过坐标变换将数据表示从“稠密”变为“稀疏”（出现 0 值），从而直观演示数据降维的数学原理。

```
import numpy as np
# 原始数据的坐标(在标准基下)
# [2,2] 意味着两个特征值都很大,且高度相关
c_old = np.array([2,2])

# 定义新基矩阵 P(过渡矩阵)
# 列向量代表新的坐标轴方向
# 第一列 [1,1] 是主方向, 第二列 [-1,1] 是垂直方向
P = np.array([[1,-1],
               [1,1]])

# 执行基变换
# 数学原理: c_old = P * c_new => c_new = P_inverse * c_old
# 使用求解线性方程组的方式, 自动计算 P 的逆并求解
c_new = np.linalg.solve(P, c_old)

print(f"【变换前】原始坐标:{c_old}")
print(f"【变换后】新坐标:{c_new}")
```

运行结果：

```
【变换前】原始坐标:[2 2]
【变换后】新坐标:[2. 0.]
```

程序运行结果显示新坐标为 $\mathbf{c}_{new} = [2, 0]^T$ ，结果显示，在切换至顺应数据分布的新基后，原本需

要两个维度描述的信息被集中于第一个分量中，第二个分量变为0。这一数值演化过程客观揭示了基变换在人工智能特征压缩中的物理意义：通过选择合适的正交基，可以实现数据信息的集中表达，从而通过剔除零值或近零值维度达到降维的目的。

1.2 范数理论与数据度量

在人工智能算法设计中，量化模型预测偏差、评估参数空间复杂度以及计算样本间的相似性是模型优化的核心环节。为了回答“向量或数据的大小是多少？”和“两个向量的差异有多大？”这类问题，数学上引入了范数向量空间的测度工具，提供了衡量向量“量级”的统一数学框架；而距离则用于表达特征空间中不同数据对象间的相对差异。

1.2.1 范数与无穷范数

在向量空间中，范数是将向量映射到非负实数的函数，用以定量描述向量的“大小”或“长度”。在金融量化分析任务中，范数常用于评估交易信号的偏差强度以及损失函数的风险权重。

(1) L_1 范数

L_1 范数是指向量中各个元素绝对值之和，记为 $\|\mathbf{x}\|_1$ 。对于 n 维实数向量 $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ ，其 L_1 范数定义为：

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$$

其中， $|x_i|$ 表示向量第 i 个分量的绝对值。在几何上，该范数对应于曼哈顿距离。由其数学性质可知， L_1 范数对小幅波动具有恒定的梯度，在特征选择任务中具有诱导稀疏解的特性。

(2) L_2 范数

L_2 范数是指向量中各个元素平方和的算术平方根，记为 $\|\mathbf{x}\|_2$ 。该范数定义了欧几里得空间中的标准测度，表达为：

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{\mathbf{x}^T \mathbf{x}}$$

L_2 范数的平方是构建均方误差 (mean squared error, MSE) 损失函数的核心。对于预测误差向量 $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$ ，其 MSE 表达为：

$$\text{MSE} = \frac{1}{n} \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

其中， $\mathbf{y}$ 为真实观测向量， $\hat{\mathbf{y}}$ 为模型预测向量。该形式对大偏差分量具有显著的惩罚效应，适用于要求参数空间平滑的回归任务。

(3) 无穷范数

无穷范数 (L_∞ 范数) 是指向量中所有元素绝对值的最大值，记为 $\|\mathbf{x}\|_\infty$ 。其定义为：

$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

从风险控制角度分析，无穷范数反映了系统在所有特征维度上的最大单点偏差。在金融风控或对抗样本防御中，该范数用于界定最差情况下的误差边界。

案例 1-4：模型预测误差的风险评估

在训练一个 AI 模型(如房价预测或股票预测)时,得到了一组“预测误差向量”。该向量包含了模型在 4 个测试样本上的预测偏差。本案例利用不同的范数来从不同角度评估模型的表现。

```
import numpy as np
from numpy.linalg import norm
# 假设误差向量 error = y_pred - y_true
# 包含了模型在 4 个样本上的预测偏差
# 样本 3 的偏差特别大(-5.0), 属于异常值
error_vector = np.array([0.1, 0.2, -5.0, 0.1])

print(f"误差向量:{error_vector}")

# L1 范数:总体绝对误差
# 物理意义:所有错误累加起来有多大?
n1 = norm(error_vector,1)
print(f"L1 范数 (总体误差累积):{n1:.2f}")

# L2 范数:误差能量
# 物理意义:误差的平方力度,对大误差非常敏感
n2 = norm(error_vector,2)
print(f"L2 范数 (误差能量/欧氏距离):{n2:.2f}")

# 无穷范数:最大风险
# 物理意义:模型犯的最严重的那个错误是多少? (安全边界)
n_inf = norm(error_vector,np.inf)
print(f"无穷范数(最大单点风险):{n_inf:.2f}")
```

运行结果:

```
误差向量:[0.1  0.2 -5.   0.1]
L1 范数(总体误差累积):5.40
L2 范数(误差能量/欧氏距离):5.01
无穷范数(最大单点风险):5.00
```

L_1 范数(5.40)反映了总体偏差的绝对值总和。 L_2 范数(5.01)虽然数值上与 L_∞ 接近,但它由误差平方和开方得出。这个结果表明,误差向量的总能量几乎完全来自于异常值 -5.0, 因为 $(-5.0)^2 = 25$, 而其他误差平方和仅为 0.06。 L_2 范数常用于衡量整体的、平滑的误差。 L_∞ 范数直接捕捉到了向量中那个最大的异常值。在人工智能安全领域,如果无穷范数过大,说明模型在某些特定样本上存在“严重失效”的风险,这是平均值(L_1/L_2)可能掩盖的信息。

通过对比可以看出,范数不仅仅是计算长度,更是评估模型好坏的不同标尺,它们反映了模型在“总体性能”“大错误惩罚力度”和“最坏情况风险”等不同维度上的表现。

1.2.2 范数正则化

在训练高维预测模型时,常面临一个两难困境:模型参数太少会导致模型欠拟合,参数太多则会记住数据中的噪声,导致模型过拟合。范数正则化是解决这一问题的核心数学工具。通过在原始损失函数中引入关于参数向量 w 的范数惩罚项,可以引导模型在最小化经验风险的同时,降低结构风险,从而提升模型的泛化能力。

目标函数的一般形式为:

$$L_{\text{total}} = J(\mathbf{w}) + \lambda \|\mathbf{w}\|_p$$

其中, $J(\mathbf{w})$ 为原始损失函数, λ 为正则化强度系数, $\|\mathbf{w}\|_p$ 为正则化惩罚项。

不同的范数作为惩罚项时, 对模型参数 $\mathbf{w}$ 的压缩方式截然不同。

1. L_2 正则化 (Ridge 回归)

L_2 正则化是指在损失函数中加入权重的 L_2 范数平方项 ($\lambda \|\mathbf{w}\|_2^2$)。 L_2 范数在参数空间中的等值线(约束区域)是一个圆(高维空间为超球体), 如图 1-6 所示。这种正则化倾向于让所有权重参数 $\mathbf{w}$ 的分量都按比例变小, 即权重衰减, 但不会严格减小到 0。它引导模型变得“平滑”, 适用于防止过拟合, 但无法实现特征选择。

L_2 正则化倾向于让权重参数 $\mathbf{w}$ 的各个分量都变小(权重衰减), 但不会减小到严格为 0。它引导模型变得“平滑”, 适用于防止过拟合, 但无法进行特征选择。

(2) L_1 正则化 (Lasso 回归)

L_1 正则化是指在损失函数中加入权重的 L_1 范数项 ($\lambda \|\mathbf{w}\|_1$)。 L_1 范数的等值线是一个菱形(或正多胞形), 其角点(顶点)尖锐且正好位于坐标轴上。由于菱形的顶点比平面其他部分更“突出”, 损失函数的等高线在接触约束区域时, 极大概率会优先接触到这些尖锐的顶点。一旦最优解落在坐标轴上, 就意味着某些维度的权重系数严格为 0, 如图 1-7 所示。

L_1 正则化具有稀疏诱导性。它能强行将不重要的特征权重压缩为 0, 从而实现自动特征选择, 有助于构建更简洁、可解释性更高的模型。

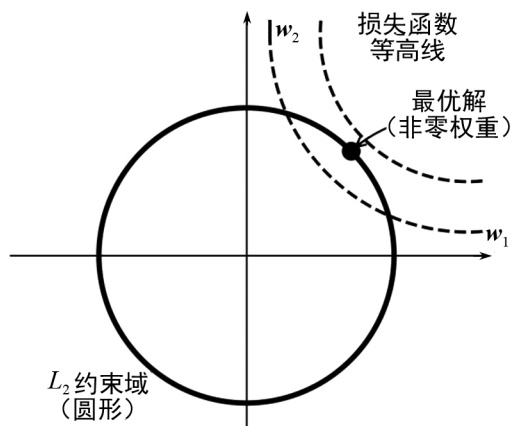


图 1-6 L_2 正则化(平滑解)

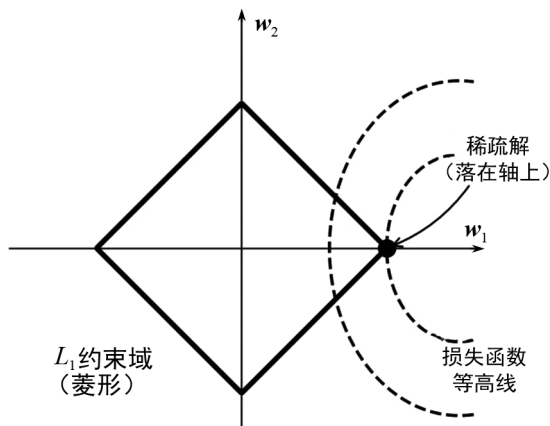


图 1-7 L_1 正则化(稀疏解)

案例 1-5: L_1 与 L_2 正则化剔除成绩无效特征

本案例构建一个预测“学生期末成绩”的线性模型, 特征空间中混入了“学号尾号”与“鞋码”这两个完全无关的噪声特征。假设当前权重向量为 $[5.0, 4.0, 0.1, -0.2, 3.0]$ (依次对应特征: 学习时长、出勤率、学号尾号、鞋码、作业分), 模型错误地利用噪声进行预测。本案例将通过模拟单步参数更新, 直观对比 L_1 正则化的“软阈值截断”与 L_2 正则化的“权重衰减”机制, 验证 L_1 范数如何将微小的噪声权重强制压缩为 0, 从而实现自动特征选择。

```

import numpy as np
# 定义特征名称和当前的过拟合权重
feature_names = np.array(["学习时长","出勤率","学号尾号","鞋码","作业分"])
w = np.array([5.0,4.0,0.1,-0.2,3.0])

print("--- 初始模型权重 ---")
for name,val in zip(feature_names,w):
    print(f"{name}:{val}")

# 设定正则化强度系数(学习率 * lambda)
penalty_strength = 0.3

# 模拟 L2 正则化更新(权重衰减)
# 数学原理: w_new = w * (1 - strength)
# 所有特征权重按比例缩小,保留信息但降低幅度
w_l2 = w * (1 - 0.1) # 假设衰减系数为 0.1
print(f"\n[L2 Ridge] 更新后权重 (平滑但不稀疏):")
print(np.round(w_l2,2))

# 模拟 L1 正则化更新(软阈值截断)
# 数学原理: w_new = sign(w) * max(|w| - strength,0)
# 小于阈值的噪声权重直接归零,大的权重保留
w_l1 = np.sign(w) * np.maximum(np.abs(w) - penalty_strength,0)

print(f"\n[L1 Lasso] 更新后权重 (稀疏特征选择):")
# 打印非零特征,模拟特征筛选过程
for name,val in zip(feature_names,w_l1):
    status = "保留" if val != 0 else "-> [已剔除]"
    print(f"{name}:{val:.1f} \t {status}")

```

运行结果:

```

--- 初始模型权重 ---
学习时长: 5.0
出勤率: 4.0
学号尾号: 0.1
鞋码: -0.2
作业分: 3.0

[L2 Ridge] 更新后权重 (平滑但不稀疏):
[ 4.5   3.6   0.09 -0.18  2.7 ]

[L1 Lasso] 更新后权重 (稀疏特征选择):
学习时长: 4.7 保留
出勤率: 3.7 保留
学号尾号: 0.0 -> [已剔除]
鞋码: -0.0 -> [已剔除]
作业分: 2.7 保留

```

可以看到, L_2 正则化后,“学号”和“鞋码”的权重虽然变小了(0.09, -0.18),但依然存在。模型仍然在利用学号和鞋码去预测成绩,这显然是不合理的。 L_1 正则化后,由于“学号”和“鞋码”的初始权重绝对值小于惩罚阈值(0.3),它们被直接强行置为 0,被剔除。

1.2.3 距离度量

如果说范数衡量的是一个向量的“绝对大小”，那么距离衡量的就是两个向量之间的“相对差异”。在人工智能中，判断两张人脸是否属于同一个人、两篇文章的话题是否相似，本质上都是在计算高维空间中两个数据点之间的距离。

在数学上，常见的距离度量通常通过闵可夫斯基距离定义：

$$d(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

对于 n 维实向量空间 $\mathbf{R}^n$ 中的两个向量 x 与 y ，根据参数 p 的不同取值，衍生出三种基础的距离度量。

(1) 欧氏距离

欧氏距离是欧几里得空间中最基本的距离度量，对应于闵可夫斯基距离中 $p=2$ 的情形。它是线性代数中 L_2 范数的直接应用。

对于 $\mathbf{R}^n$ 空间中的两个向量 $\mathbf{x}$ 和 $\mathbf{y}$ ，欧氏距离定义为向量差的 L_2 范数，即各分量差的平方和的算术平方根：

$$d_{\text{euc}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

欧氏距离对特征空间的各个维度赋予等同权重，但由于平方运算的存在，它对数值较大的离群点或噪声较为敏感，适用于 K-Means 聚类算法、KNN 分类算法。

(2) 曼哈顿距离

曼哈顿距离对应于闵可夫斯基距离中 $p=1$ 的情形，也被称为 L_1 距离或出租车几何距离。

曼哈顿距离定义为两个向量在各个维度上坐标差的绝对值之和。其数学表达式为：

$$d_{\text{man}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^n |x_i - y_i|$$

曼哈顿距离的几何等值面呈现为超正方体，它相比欧氏距离对离群点具有更好的鲁棒性，在处理稀疏数据（如文本词频统计）或高维特征贡献差异巨大的场景中表现良好。

(3) 切比雪夫距离

切比雪夫距离是闵可夫斯基距离当 p 趋向于无穷大（ $p \rightarrow \infty$ ）时的极限形式，对应于 L_∞ 范数。

切比雪夫距离定义为两个向量在所有维度上坐标差的绝对值中的最大值。其数学表达式为：

$$d_{\text{cheb}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_\infty = \max_i (|x_i - y_i|)$$

该度量方式表明，两个数据对象之间的整体差异完全取决于差异最大的那个特征维度，适用于极值起决定性作用的特殊场景。

三种距离度量在数值上严格遵循距离不等式链： $d_{\text{cheb}} \leq d_{\text{euc}} \leq d_{\text{man}}$ 。这反映了不同范数对特征差异的聚合程度： L_∞ 仅取最大偏差， L_2 对偏差进行均方聚合，而 L_1 则是偏差的直接线性累加。

案例 1-6：基于闵可夫斯基距离的用户偏好判定

在推荐系统中，判断新用户属于哪一类人群通常基于“最近邻”思想：距离谁最近，就与谁兴趣最相似。

现有三位用户对三类电影（动作、爱情、科幻）的评分向量（0~5 分）：

- 用户 A(硬汉型):[5, 1, 5]
- 用户 B(浪漫型):[1, 5, 1]
- 新用户 C(未知):[4, 2, 4]

本例分别使用曼哈顿距离、欧氏距离和切比雪夫距离，来量化 C 与 A、B 之间的差异。

```
import numpy as np
from numpy.linalg import norm

# 定义用户评分向量
# 向量含义:[动作片, 爱情片, 科幻片]
user_A = np.array([5,1,5])
user_B = np.array([1,5,1])
user_C = np.array([4,2,4])

print(f"用户 A:{user_A}")
print(f"用户 B:{user_B}")
print(f"用户 C:{user_C}\n")

# 计算差向量
diff_AC = user_A - user_C
diff_BC = user_B - user_C

# 计算三种不同的距离
# 曼哈顿距离(p=1):绝对值之和
d1_AC = norm(diff_AC,1)
d1_BC = norm(diff_BC,1)

# 欧氏距离(p=2):直线距离
d2_AC = norm(diff_AC,2)
d2_BC = norm(diff_BC,2)

# 切比雪夫距离(p=∞):最大单一维度偏差
d_inf_AC = norm(diff_AC,np. inf)
d_inf_BC = norm(diff_BC,np. inf)

print("--- 距离计算结果 (数值越小越相似) ---")
print(f"曼哈顿距离 (L1) :C-A = {d1_AC:. 1f}, C-B = {d1_BC:. 1f}")
print(f"欧氏距离 (L2) :C-A = {d2_AC:. 2f}, C-B = {d2_BC:. 2f}")
print(f"切比雪夫距离(L∞) :C-A = {d_inf_AC:. 1f}, C-B = {d_inf_BC:. 1f}")
```

运行结果:

```
用户 A:[5 1 5]
用户 B:[1 5 1]
用户 C:[4 2 4]

--- 距离计算结果 (数值越小越相似) ---
曼哈顿距离 (L1) :C-A = 3.0, C-B = 9.0
欧氏距离 (L2) :C-A = 1.73, C-B = 5.20
切比雪夫距离(L∞) :C-A = 1.0, C-B = 3.0
```

可以发现，无论采用哪种距离度量，用户 C 与用户 A 的距离都显著小于与用户 B 的距离。这表明在特征空间中，新用户 C 更接近用户 A，推荐系统应向 C 推荐 A 喜欢的电影。

1.3 内积空间与相似性

在向量空间的结构中,若说范数度量了向量自身的“长度”或“大小”,那么内积则量化了两个向量之间的“方向差异”与“投影关系”。引入内积运算后,向量空间转化为内积空间,为解决正交性变换、特征相似度度量以及流形学习等问题提供了代数基础。该理论构成了卷积神经网络(CNN)中的特征提取、支持向量机(SVM)中的核映射以及主成分分析(PCA)中坐标投影的数学准则。

1.3.1 内积与几何投影

内积,在欧几里得空间中通常被称为点积,是将两个向量映射为一个标量的运算。它不仅是代数上的乘法累加,更深刻地反映了两个向量在方向上的相关程度。

(1) 代数定义

设 $\mathbf{R}^n$ 空间中有两个向量 $\mathbf{x}=[x_1, x_2, \dots, x_n]^T$ 和 $\mathbf{y}=[y_1, y_2, \dots, y_n]^T$ 。其代数内积(点积)定义为对应分量乘积之和,记作 $\langle \mathbf{x}, \mathbf{y} \rangle$ 或 $\mathbf{x} \cdot \mathbf{y}$:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^n x_i y_i$$

(2) 几何投影与正交性

由余弦定理可知,内积建立了代数运算与几何夹角之间的映射关系:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{x}\| \|\mathbf{y}\| \cos \theta$$

其中, $\|\mathbf{x}\|$ 和 $\|\mathbf{y}\|$ 分别是向量的模(L_2 范数), θ 是两个向量之间的夹角($\theta \in [0, \pi]$)。

内积最直观的物理意义是“投影”。将几何公式改写为:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{y}\| (\|\mathbf{x}\| \cos \theta)$$

括号中的项 $\|\mathbf{x}\| \cos \theta$ 具有明确的几何含义:它代表了向量 $\mathbf{x}$ 在向量 $\mathbf{y}$ 方向上的标量投影。内积 $\langle \mathbf{x}, \mathbf{y} \rangle$ 的数值大小,等于 $\mathbf{x}$ 在 $\mathbf{y}$ 方向上的投影长度乘以 $\mathbf{y}$ 自身的长度。

- 当 $\mathbf{x}$ 与 $\mathbf{y}$ 方向基本一致(夹角为锐角)时,投影为正,内积为正,表示两者存在正相关的“合力”。
- 当 $\mathbf{x}$ 与 $\mathbf{y}$ 相互垂直(夹角为 90°)时, $\cos \theta = 0$, 投影长度为零,内积为零。这种状态称为正交,表示两个向量线性无关,彼此不包含对方的信息。
- 当 $\mathbf{x}$ 与 $\mathbf{y}$ 方向相反(夹角为钝角)时,投影与 $\mathbf{y}$ 反向,内积为负,表示方向相反。

(3) 相似性度量与特征检测

在人工智能中,内积不仅仅是几何上的投影,更是一种衡量两个向量“结构相似程度”的工具。通常将向量视为一个固定的“基准模式”或“过滤器”(例如 CNN 中的卷积核),而将 $\mathbf{x}$ 视为输入的“数据信号”。内积 $\langle \mathbf{x}, \mathbf{y} \rangle$ 的计算过程,实质上是对数据进行一次扫描与匹配。

- 高度匹配:当内积数值很大(为正)时,说明数据 $\mathbf{x}$ 的特征与基准 $\mathbf{y}$ 的方向高度一致,即数据中包含了该模式。
- 无关(正交):当内积接近 0 时,说明数据 $\mathbf{x}$ 与基准 $\mathbf{y}$ 相互垂直,数据中不包含该特定的特征信息。
- 相反(抑制):当内积为负时,说明数据 $\mathbf{x}$ 呈现出与基准 $\mathbf{y}$ 相反的特征。

这种机制是卷积神经网络提取图像边缘、纹理等特征的数学原理——内积越大，特征被“激活”的程度越高。

案例 1-7：模拟卷积神经网络中的特征检测

在计算机视觉中，卷积核本质上就是一个包含特定权重的向量，边缘检测器通过内积运算识别图像中的灰度跃变。假设有一个专门用于检测边缘特征的卷积核（模式向量），以及两个来自图像不同区域的像素信号向量。

- 卷积核： $[1, 1, -1, -1]$ （模拟从亮到暗的边缘模式）。
- 图像区域 A： $[10, 10, 0, 0]$ （也是从亮到暗，与模式匹配）。
- 图像区域 B： $[5, 5, 5, 5]$ （平坦区域，无边缘）。

本例利用计算卷积核与两个代表不同局部区域的像素向量的内积，通过比较输出激活值的大小，量化特定几何特征的识别效能。

```
import numpy as np
# 定义卷积核
# "边缘检测器", 模式为: 前一半正, 后一半负
kernel = np.array([1,1,-1,-1])

# 定义图像的局部像素向量
# img_edge:明显的边缘(前亮后暗),与卷积核模式一致
img_edge = np.array([10,10,0,0])
# img_flat:平坦区域(颜色一致),没有边缘特征
img_flat = np.array([5,5,5,5])

# 利用内积进行"卷积"运算(特征提取)
# 内积值越高,表示该区域与卷积核的模式越匹配
activation_edge = np.dot(img_edge,kernel)
activation_flat = np.dot(img_flat,kernel)

print(f"卷积核模式 w:{kernel}")
print(f"图像区域 A (边缘):{img_edge} -> 激活值 (内积):{activation_edge}")
print(f"图像区域 B (平坦):{img_flat} -> 激活值 (内积):{activation_flat}")

# 判定逻辑
threshold = 10.0
if activation_edge > threshold:
    print("\n 结论: 在区域 A 检测到边缘特征!")
else:
    print("\n 结论: 区域 A 无明显特征。")
```

运行结果：

```
卷积核模式 w:[ 1  1 -1 -1]
图像区域 A (边缘):[10 10  0  0] -> 激活值 (内积):20
图像区域 B (平坦):[5 5 5 5] -> 激活值 (内积):0

结论: 在区域 A 检测到边缘特征!
```

上面的结果显示在区域 A 检测到边缘特征。

区域 A(20) 内积值很大。因为图像向量的变化趋势与卷积核的方向高度一致(投影长度大), 产生了强烈的激活信号。

区域 B(0) 内积值为 0。尽管区域 B 的像素有亮度, 但它是平坦的, 与边缘检测器正交。这直观展示了内积如何作为“滤波器”, 自动过滤掉不感兴趣的信号。

1.3.2 相似性度量

内积受向量模长的影响, 无法直接作为纯粹的相似度指标。例如, 两个用户对电影的评分向量可能是“老好人”和“批评家”。虽然喜好完全一致, 但内积值会差 100 倍, 导致错误的结论。

为了消除这种量纲差异, 需要对内积进行归一化处理, 从而将向量“拉到”单位球面上, 只关注方向信息。由此引出余弦相似度和皮尔逊相关系数。

(1) 余弦相似度

余弦相似度是通过测量两个向量在多维空间中夹角的余弦值来判断其相似性的度量。它通过模长归一化, 消除了向量模长的影响, 仅保留了方向信息。在自然语言处理(NLP)和推荐系统等高维特征空间中, 余弦相似度是衡量相似性的常用指标。

给定两个非零向量 $\mathbf{x}$ 和 $\mathbf{y}$, 它们的余弦相似度定义为内积与模长乘积的比值:

$$\text{sim}_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

余弦相似度的取值范围为 $[-1, 1]$ 。值越接近 1, 方向越相似; 值为 0, 表示向量正交(相互独立); 值越接近 -1, 方向越相反。余弦相似度对向量的整体数值非常敏感。例如, 如果所有向量的分量都为正数, 那么余弦相似度的值永远为正, 无法区分“负相关”关系。

(2) 皮尔逊相关系数

皮尔逊相关系数(记为 ρ 或 r) 用于度量两个变量之间的线性相关程度, 它解决了余弦相似度对数据“偏差”或“基准”敏感的问题。

在推荐系统中, 不同用户的评分基准(bias)存在差异。皮尔逊相关系数通过对向量执行中心化处理(减去均值), 消除了个体偏好基准的影响:

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

其中, $\bar{x}$ 、 $\bar{y}$ 分别为向量分量($\mathbf{x}$ 、 $\mathbf{y}$)的算术平均值。皮尔逊相关系数可视为中心化后的余弦相似度, 反映了数据波动趋势的同步性。

案例 1-8: 相似度消除用户评分偏差的影响

在推荐系统中, 不同用户的打分习惯存在差异。

用户 A(“老好人”) 打分普遍偏高, 均值为 4.0。

用户 B(“批评家”) 打分普遍偏低, 均值为 1.0。

尽管绝对分数差异大, 但两人对项目的喜好趋势完全一致。本例验证哪种度量能有效捕捉这种内在趋势的一致性。

```

import numpy as np
from numpy.linalg import norm
print("=== 相似度度量对比验证 ===")
user_a = np.array([5.0, 5.0, 4.0, 3.0, 3.0])
user_b = np.array([2.0, 2.0, 1.0, 0.0, 0.0])

def cosine_similarity(v1, v2):
    # 避免除以 0
    return np.dot(v1, v2) / (norm(v1) * norm(v2))

def pearson_correlation(v1, v2):
    # 计算中心化向量的余弦相似度
    v1_centered = v1 - np.mean(v1)
    v2_centered = v2 - np.mean(v2)
    return cosine_similarity(v1_centered, v2_centered)

# 欧氏距离
dist_euc = norm(user_a - user_b)

# 余弦相似度
sim_cos = cosine_similarity(user_a, user_b)

# 皮尔逊相关系数
corr_pearson = pearson_correlation(user_a, user_b)

print(f"欧氏距离: {dist_euc:.4f} ")
print(f"余弦相似度: {sim_cos:.4f} ")
print(f"皮尔逊相关系数: {corr_pearson:.4f} ")

```

运行结果:

```

=== 相似度度量对比验证 ===
欧氏距离:      6.7082
余弦相似度:    0.8729
皮尔逊相关系数: 1.0000

```

欧氏距离值较大(6.71)误认为用户差异巨大;余弦相似度值高(0.87)但仍未达到完美相关;皮尔逊相关系数为 1.00,准确反映了虽然两个用户打分绝对值有偏差(均值分别为 4.0 和 1.0),但其对项目的喜好趋势是完全同步的。

这说明两人的“相对偏好”完全一致,皮尔逊相关系数因此常用于修正推荐系统中的用户评分偏差。

1.3.3 正交向量组与标准正交基

在 1.1 节中我们讨论了基向量必须线性无关,在 1.3.1 节中我们定义了内积。当这两个概念结合时,向量空间中便形成了正交基这种重要的结构。在人工智能特征工程与信号处理任务中,消除特征间的冗余性与多重共线性是提升模型鲁棒性的关键。数学上,这一目标的实现主要依赖于向量组的正交变换。

(1) 正交向量组

在欧几里得空间中,如果一组非零向量两两之间的内积均为零,则称该组向量为正交向量组。

设 $\mathbf{R}^n$ 空间中有一组非零向量 $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\}$ 。若对于任意的 $i \neq j$, 均满足 $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$, 则称这组向量为正交向量组。由代数性质可知, 正交向量组必线性无关。从几何投影角度分析, 正交向量组中的每一个向量在其他向量方向上的投影分量皆为零。在 AI 自动特征提取任务中, 这表示各特征维度所承载的信息互不重叠, 从而保证了特征表达的简洁性。

(2) 标准正交基

一组基 $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ 满足正交性且各向量模长均为 1, 则称为标准正交基。其代数判定准则表达为:

$$\langle \mathbf{e}_i, \mathbf{e}_j \rangle = \begin{cases} 1 & i=j(\text{单位化}), \\ 0 & i \neq j(\text{正交性}). \end{cases}$$

标准正交基简化了投影计算, 向量 $\mathbf{x}$ 在该基下的第 i 个坐标可直接通过内积 $\langle \mathbf{x}, \mathbf{e}_i \rangle$ 获取。以标准正交基作为坐标轴, 不同特征维度之间的协方差恒为 0。这种特征解耦特性能够有效简化神经网络的收敛路径, 在生成对抗网络 (GAN) 或变分自编码器 (VAE) 的隐空间构建中起着至关重要的作用。

案例 1-9: 标准正交基与特征的去相关性

在训练机器学习模型时, 如果输入特征 (房屋面积、房屋估价) 之间存在多重共线性 (即高度相关), 模型会变得不稳定, 且难以解释特征的重要性。主成分分析 (PCA) 通过将数据投影到一个由标准正交基构成的子空间, 从根本上消除了这种相关性, 这是特征工程中的关键一步。

本例验证标准正交变换如何实现特征的去相关。数据的协方差矩阵 ($\mathbf{C}$) 用于衡量特征之间的相关性。如果特征不相关, 则 $\mathbf{C}$ 是一个对角矩阵 (非对角线元素为 0)。标准正交基的作用是找到一个变换矩阵 $\mathbf{P}$ ($\mathbf{P}^T \mathbf{P} = \mathbf{I}$), 使得新特征空间的协方差矩阵 $\mathbf{P}^T \mathbf{C} \mathbf{P}$ 成为对角矩阵。

```
import numpy as np
# 创建两个高度相关的原始特征(模拟多重共线性)
# 假设 X1 是房屋面积,X2 是房屋估价,两者高度相关
np.random.seed(42)
X1 = np.arange(10) + np.random.normal(0,0.5,10)
X2 = 3 * X1 + np.random.normal(0,1.0,10) # X2 约等于 3 倍 X1 加噪声

# 中心化处理: 减去均值,是 PCA 的第一步
X_data = np.vstack((X1,X2)).T
X_centered = X_data - np.mean(X_data,axis=0)

# 计算原始特征的协方差矩阵
cov_original = np.cov(X_centered,rowvar=False)

print("\n--- 原始特征的协方差矩阵 ---")
# 非对角线元素代表 X1 与 X2 的协方差
print(cov_original.round(2))

# 定义一个标准正交变换矩阵 P(模拟 PCA 找到的主成分方向)
# P 的列向量 e1 和 e2 是标准正交基,用于构造新的坐标系
theta = np.deg2rad(70)
P = np.array([[np.cos(theta),- np.sin(theta)],
```

```
[np. sin(theta), np. cos(theta)]]

# 执行标准正交变换:  $Y = X\_centered @ P$ 
Y_transformed = X_centered @ P

# 计算变换后新特征的协方差矩阵
cov_transformed = np. cov(Y_transformed,rowvar=False)

print("\n--- 标准正交基变换后的协方差矩阵 ---")
# 理想情况下,非对角线元素应接近于 0(对角矩阵)
print(cov_transformed. round(2))
```

运行结果:

```
--- 原始特征的协方差矩阵 ---
[[ 9.23 27.26]
 [27.26 81.03]]

--- 标准正交基变换后的协方差矩阵 ---
[[90.152.19]
 [ 2.19  0.11]]
```

原始特征的协方差为 27.26, 表明特征高度相关。通过应用由标准正交基构成的矩阵 P 进行坐标变换后, 新特征空间的协方差(非对角线元素)降至 2.19。尽管 2.19 尚未达到理想的 0(这通常意味着 P 只是一个近似的最佳旋转), 但相对于原始的 27.26 已经实现了显著的统计不相关性(去相关)。

这证明了标准正交基是实现特征解耦、解决多重共线性问题的关键数学工具, 是许多 AI 算法, 特别是主成分分析(PCA)和奇异值分解(SVD)的核心理论基础。

第 2 章

矩阵分析与线性变换

本章导读

本章聚焦于矩阵分析的核心理论及其在人工智能领域中的功能性应用。矩阵不仅是高维数据的存储载体，更是实现空间线性变换的基本算子。本章首先从几何视角审视矩阵乘法、逆矩阵及行列式，阐述线性映射如何实现特征空间的转换，以及行列式对空间体积缩放的量化作用。随后，探讨具有特殊代数结构的矩阵及其在特定算法中的约束功能，包括对称矩阵对双向关系的表征、正交矩阵对空间旋转与数值稳定性的维持，以及邻接矩阵与拉普拉斯矩阵对非欧几里得拓扑结构的数学刻画。最后，通过引入矩阵秩的概念，揭示数据特征间的线性相关性，并阐述低秩逼近在信息压缩、去噪与降维任务中的底层支撑作用。通过本章的学习，读者将能够建立起从空间变换与特征提取的角度理解线性代数的思维框架，为后续研究机器学习模型、图神经网络 (graph neural network, GNN) 及流形学习奠定严谨的数学基础。

学习指导

通过本章学习，你将：

- 理解与掌握矩阵乘法的线性映射视角，能够从空间变换的角度分析数据特征的映射逻辑。
- 掌握逆矩阵与广义逆矩阵的代数定义，理解行列式在量化特征空间体积缩放中的几何意义。
- 熟练运用对称矩阵与正交矩阵的数学特性，理解其在表征关联度与维持坐标变换稳定性中的功能。
- 理解邻接矩阵与拉普拉斯矩阵的构造原理，并掌握其在图神经网络拓扑结构建模中的应用。
- 掌握矩阵秩的定义及其与数据线性相关性的内在联系，能够识别高维数据集中的信息冗余。
- 理解矩阵低秩逼近的数学原理，并运用该理论解决信息压缩、数据去噪与特征提取等实际应用问题。

2.1 矩阵运算与几何意义

矩阵是线性代数中最核心的数学对象，它是数据（特征向量）的二维组织形式，更是线性变换的数学载体。在人工智能算法中，矩阵不仅用于结构化存储特征数据，更作为线性算子执行特征空间的映射与维度转换。

2.1.1 矩阵乘法与线性映射视角

矩阵乘法是执行特征空间转换的核心代数运算。从函数映射角度分析，矩阵乘法定义了一个从定义域空间向值域空间的线性映射。

(1) 代数定义

设 $A \in \mathbf{R}^{m \times k}$ 为映射矩阵， $B \in \mathbf{R}^{k \times n}$ 为包含 n 个特征向量的矩阵，则乘积矩阵 $C = AB \in \mathbf{R}^{m \times n}$ 的第 i 行第 j 列元素 c_{ij} 定义为矩阵 A 的第 i 个行向量与矩阵 B 的第 j 个列向量的内积：

$$c_{ij} = \sum_{l=1}^k a_{il} b_{lj}$$

该公式成立的前提是左矩阵的列数与右矩阵的行数严格对齐，这反映了两个特征空间维度的相容性。

(2) 几何意义与线性变换

由几何性质可知，当矩阵 A 作用于向量 x （即 $y = Ax$ ）时， A 执行了线性变换。

- 结果向量 y 实质上是矩阵 A 的各列向量以 x 的分量为权重的线性组合。这意味着变换后的向量被限制在 A 的列空间内。
- 矩阵变换包含了对原始特征空间的旋转、缩放以及剪切。在自动驾驶轨迹预测任务中，这种变换用于描述车辆在不同时刻坐标系之间的位姿演变。

(3) 线性映射在神经网络中的逻辑延伸

在深度神经网络的线性层中，输入特征 x 与权重矩阵 W 的运算 $z = Wx + b$ 构成了模型的基础单元。映射矩阵 W 充当了特征提取器，将原始输入映射到更高维或更低维的激活空间。通过调整矩阵的分量，模型能够改变特征在空间中的相对位置，从而实现不同类别的线性可分。

此外，在主成分分析（PCA）等降维技术中，矩阵乘法被用于将高维数据投影到低维正交子空间，以去除数据冗余；在信号和图像处理中，卷积运算本质上也可分解为稀疏矩阵的乘法，用于提取局部特征。因此，矩阵乘法是实现人工智能模型特征学习和数据降维的核心数学工具。

案例 2-1：校园生活状态评估中的特征变换

本案例演示神经网络如何利用矩阵乘法（ $z = Wx$ ）实现特征线性映射。即通过权重矩阵 W 的转化，将直观的行为数据（输入 x ）抽象为高层的状态特征（输出 z ）。

假设构建一个“校园生活分析模型”，将学生一天的 3 项基础行为时长映射为 2 种生活状态指数：

①输入向量 x （基础行为，单位：小时）

- x_1 自习时长：反映学习投入（如图书馆自习）。

- x_2 社团/娱乐：反映社交与休闲(如社团活动、游戏)。
 - x_3 睡眠时长：反映作息规律(如夜间休息)。
- ②输出向量 z (状态指数, 模型打分)
- z_1 “学霸”指数：综合衡量学术专注度与效率。
 - z_2 “休闲”指数：综合衡量社交活跃度与生活丰富度。

```
import numpy as np
# 定义输入特征 x(3 维向量)
# 假设某位同学的一天:自习 8 小时,社团/娱乐 2 小时,睡眠 6 小时
x = np.array([8, 0.2, 0.6, 0])

# 定义权重矩阵 W(2x3 矩阵)
# W 里的数值代表了模型定义的"价值观":不同的行为对最终状态有多大贡献
W = np.array([
    # 自习(x1)  社团/娱乐(x2)  睡眠(x3)
    [ 1.5,      -0.5,      0.5], # 第一行:计算"学霸"指数的权重
    [-0.2,      1.2,      0.3]  # 第二行:计算"休闲"指数的权重
])

# 执行线性映射 z = Wx
# 本质是:行向量(权重)与列向量(输入)的点积
z = W @ x

print(f"输入行为数据 x:{x}")
print(f"权重矩阵 W:\n{W}")
print("- " * 30)
print(f"输出状态指数 z:{z}")
```

运行结果：

```
输入行为数据 x:[8. 2. 6.]
权重矩阵 W:
[[ 1.5 -0.5  0.5]
 [-0.2  1.2  0.3]]
-----
输出状态指数 z:[14.  2.6]
```

运行结果 $z = [14.0, 2.6]$ 展示了该同学“重学业、轻社交”的特征。矩阵乘法通过加权求和机制 (Wx), 利用第一行对“自习”的高正向权重(1.5)将8小时的学习投入转化为高达14.0的“学霸”指数, 同时因“社团/娱乐”投入较少且受部分负权重抑制, 计算出的“休闲”指数仅为2.6。这一过程直观展示了线性层如何通过权重矩阵识别数据模式, 将原始行为数据成功抽象为具有语义的高级状态特征。

2.1.2 逆矩阵与广义逆矩阵

在人工智能的特征映射过程中, 矩阵乘法通常表达从输入特征空间向目标表示空间的“正向映射”。将输入数据映射为分类结果(模型推断), 或者将图像进行旋转缩放(几何变换)。

然而, 我们经常面临“逆向”的需求:

- 模型训练(反向传播): 已知预测结果和真实标签的误差, 如何反推权重的调整量?
- 数据复原: 已知被模糊或压缩过的图像, 如何尽可能还原出原始画面?

- 求解方程：已知输出结果，如何反解出输入数据？

然而，在模型参数优化、信号复原以及系统状态推断等任务中，通常需要执行上述过程的逆向求解，这一过程在代数上依赖于矩阵的逆运算。

(1) 逆矩阵的代数定义与几何性质

对于 n 阶方阵 A ，如果存在 n 阶方阵 B ，使得它们相乘的结果等于单位矩阵 I ：

$$AB = BA = I$$

那么， B 就称为 A 的逆矩阵，记作 A^{-1} 。

从几何视角分析，若矩阵 A 代表一种线性变换（如坐标系旋转或特征缩放），则 A^{-1} 描述了该变换的逆过程。逆矩阵存在的充要条件是矩阵 A 为非奇异矩阵，即其行列式 $\det(A) \neq 0$ （满秩）。

由几何性质可知，若 $\det A = 0$ ，则意味着该变换导致了空间的“维度坍塌”，即高维空间的信息被压缩至低维子空间。在这种情况下，原始空间的信息发生了不可逆的丢失，导致映射过程不可逆。

(2) 广义逆矩阵

在机器学习的工程实践中，样本数据矩阵 $X \in \mathbf{R}^{m \times n}$ 通常不是方阵（样本数 m 通常远大于特征数 n ）。对于超定方程组 $Xw = y$ ，通常不存在精确解。为了在特征空间中寻求最优近似，引入了穆尔-彭罗斯广义逆矩阵，记为 X^+ 。

(3) 线性回归中的最小二乘法

广义逆矩阵在人工智能中最经典的应用就是线性回归。

当试图训练一个模型 $Xw = y$ 来拟合数据时，由于数据存在噪声，通常无法找到一条直线完美穿过所有点（即方程组无精确解）。此时，目标转变为：找到一个权重向量 w ，使得预测误差最小。

利用广义逆矩阵，可以直接写出这个“最优解”的解析公式：

$$w = X^+ y = (X^T X)^{-1} X^T y$$

总之，逆矩阵适用于完美的方阵系统，代表精确的逆变换。广义逆矩阵适用于任意形状的矩阵，代表在无法精确求解时，寻找误差最小的“最佳妥协解”。这是机器学习算法处理现实数据的基础工具。

案例 2-2：利用广义逆矩阵求解薪资预测模型“无解”方程

本案例演示如何利用广义逆矩阵解决人工智能中最基础的线性回归问题。在现实中，数据往往充满噪声，方程组通常是“无解”的（过定方程）。本例展示如何通过广义逆找到“最佳近似解”，即训练出一个误差最小的模型参数。

假设要通过几组历史数据，训练一个简单的线性模型，用来预测工作年限与时薪的关系。

模型假设：

$$\text{时薪}(y) = w \times \text{工作年限}(x)$$

收集了 3 位员工的数据：

- 员工 A：工作 1 年，时薪 30 元。
- 员工 B：工作 2 年，时薪 50 元。
- 员工 C：工作 3 年，时薪 80 元。

将数据代入方程 $x \cdot w = y$ ：

$$\begin{cases} 1 \times w = 30 \Rightarrow w = 30, \\ 2 \times w = 50 \Rightarrow w = 25, \\ 3 \times w = 80 \Rightarrow w \approx 26.7. \end{cases}$$

显然,不存在一个单一的 w 能同时满足这三个等式。写成矩阵形式 $Xw=y$:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot w = \begin{bmatrix} 30 \\ 50 \\ 80 \end{bmatrix}$$

由于矩阵 X 是 3×1 的长矩阵,且三个方程互相矛盾,传统的逆矩阵无法求解。此时,需要使用广义逆矩阵。利用 `numpy.linalg.pinv` 计算广义逆,求出最佳权重。

```
import numpy as np
# 定义输入数据矩阵 X(3 行 1 列,代表 3 个样本的经验)
X = np.array([[1],[2],[3]])
# 定义目标标签向量 y(3 个样本对应的真实时薪)
y = np.array([30,50,80])
# 求解权重 w
# 由于 X 不是方阵,不能用 inv 求逆,必须用 pinv(伪逆/广义逆)
# 公式:w = X_dagger @ y
X_dagger = np.linalg.pinv(X)
w = X_dagger @ y

print(f"输入矩阵 X:\n{X}")
print(f"广义逆矩阵 X_dagger:\n{X_dagger}")
print("- " * 30)
print(f"计算得到的最佳权重 w:{w.item():.2f}")

# 验证模型效果
y_pred = X @ w
print(f"模型预测值:{y_pred}")
print(f"真实值 y: {y}")
```

运行结果:

```
输入矩阵 X:
[[1]
 [2]
 [3]]
广义逆矩阵 X_dagger:
[[0.07142857 0.14285714 0.21428571]]
-----
计算得到的最佳权重 w:26.43
模型预测值:[26.42857143 52.85714286 79.28571429]
真实值 y: [30 50 80]
```

上面的结果计算出的权重 $w \approx 26.43$ 。

- 对于员工 A(1 年), 预测 26.4 元(比真实少 3.6 元)。
- 对于员工 B(2 年), 预测 52.8 元(比真实多 2.8 元)。
- 对于员工 C(3 年), 预测 79.3 元(比真实少 0.7 元)。

虽然模型没有完美预测任何一个人的工资,但 $w=26.43$ 是一个让总误差平方和最小的解。

2.1.3 行列式与几何体积缩放

在传统的线性代数课程中,矩阵的行列式(determinant, 记作 $\det A$ 或 $|A|$)常被视为一个仅仅用

于判断矩阵是否可逆的数值。但在人工智能与几何机器学习中，行列式不仅用于判定矩阵的可逆性，其核心几何意义在于描述线性变换对空间测度(面积或体积)的缩放倍率。

(1) 几何意义与空间定向

当矩阵 A 作用于特征空间时，原本由标准基向量构成的几何图形将发生形变。

① 二维空间(面积缩放)

考察一个由基向量 $e_1 = (1, 0)$ 和 $e_2 = (0, 1)$ 围成的单位正方形，其面积为 1。经过矩阵 A 变换后，这两个基向量分别映射为 Ae_1 和 Ae_2 ，原本的正方形随之变为一个平行四边形。此时，行列式的绝对值即为该平行四边形的面积，如图 2-1 所示。左图为原始单位空间，右图为变换后的空间，行列式的值对应于面积的缩放倍率。

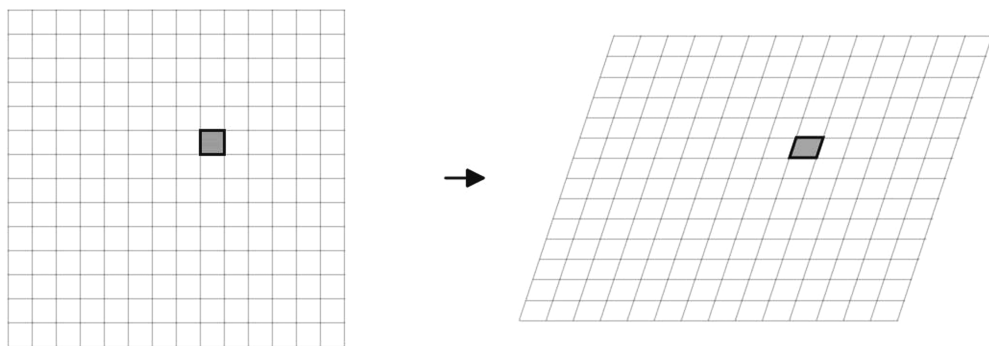


图 2-1 线性变换的几何意义

② 三维及高维空间(体积缩放)

同理，在三维空间中，基向量组成的“单位立方体”在变换后变为“平行六面体”。行列式的绝对值代表了体积的缩放倍率。

这一性质不仅适用于简单的几何图形，也适用于任意复杂的数据分布。行列式揭示了数据在经过线性层处理后，其在空间中的分布密度是变得稀疏还是稠密。

(2) 符号与空间定向

行列式的正负符号反映了线性变换对空间方向性的影响。

- 定向保持($\det A > 0$)：当行列式为正时，变换保持了基向量的相对位置关系。例如在三维空间中，若变换前的坐标系符合右手定则，变换后的坐标系依然符合右手定则。这代表空间仅发生了旋转或剪切，未发生翻转。

- 定向反转($\det A < 0$)：当行列式为负时，变换改变了空间的定向。这在几何上等价于空间发生了一次镜像反射。此时，原本的右手系会被映射为左手系，物体被翻转到了相反的方向。

虽然在体积计算中通常取绝对值，但在某些图形算法和物理模拟中，符号的正负至关重要。

(3) 奇异性与维度坍缩

结合逆矩阵的定义，行列式的几何意义能够直观解释为何行列式为零的矩阵不可逆。

当 $\det A = 0$ 时，意味着线性变换的体积缩放因子为零。在几何上，这表示变换后的空间体积完全消失。

维度坍缩表示原本 n 维的空间结构被压缩到了一个 $n-1$ 维或更低维的子空间中。例如，三维立方体被变换压缩为一个平面、一条直线甚至原点。在维度坍缩的过程中，原本空间中位置不同的多个点，被同时映射到了目标空间中的同一点(多对一映射)。这种信息的混叠导致了原始位置信息的

永久性丢失。由于无法确定变换前的原始状态，逆变换不存在。当行列式为 0 时，原本二维的平面(图 2-2 左图)被压缩成了一条一维的直线(图 2-2 右图)，导致面积归零且信息丢失，此类矩阵不可逆。

因此， $\det \mathbf{A} = 0$ 的矩阵被称为奇异矩阵。它代表了一种破坏性的、不可逆的数据压缩操作。

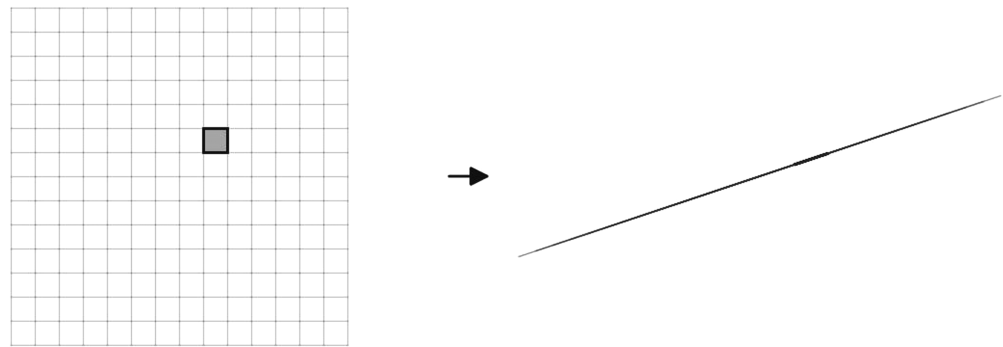


图 2-2 奇异矩阵导致的维度坍塌

(4) 雅可比行列式与非线性变换

在机器学习的生成模型(如标准化流模型 Normalizing Flows 或变分自编码器 VAE)中，网络的主要任务是构建从已知简单分布(如高斯分布)到复杂数据分布(如图像空间)的映射。为了处理这种非线性变换中的概率密度问题，需要引入雅可比行列式。

① 局部线性化与体积缩放

线性变换对空间的缩放是全局一致的，而神经网络中的非线性变换在空间不同位置的变形程度各不相同。

对于一个多维非线性函数，其在某一点 x 处的偏导数构成了雅可比矩阵 $\mathbf{J}$ 。从几何视角看，雅可比矩阵描述了函数在 x 点附近的局部线性近似。

雅可比矩阵的行列式 $\det \mathbf{J}$ 量化了空间在该点附近的局部体积缩放倍率。若 $|\det \mathbf{J}| > 1$ ，说明该局部的体积发生了膨胀。若 $|\det \mathbf{J}| < 1$ ，说明该局部的体积发生了收缩。这种逐点的体积精确计算，是理解非线性网络几何行为的关键。

在非线性变换(如流模型)中，网格发生扭曲。但在局部微小的区域(如图 2-3 中深色方块所示)，变换仍可近似为线性，其面积变化率由该点的雅可比矩阵的行列式决定。

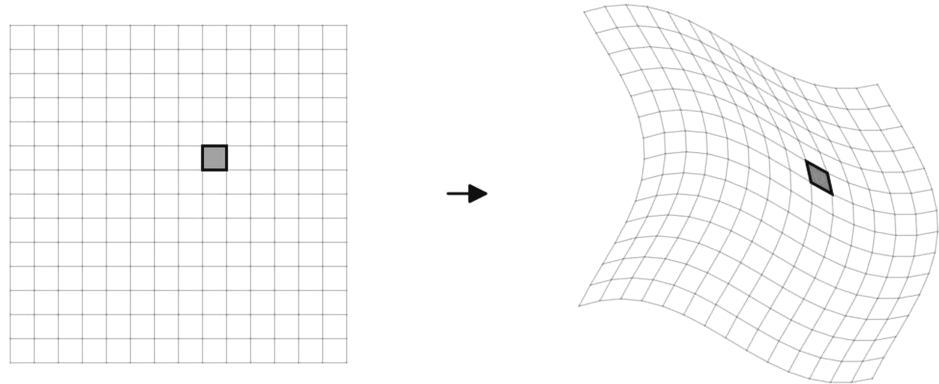


图 2-3 非线性变换与雅可比行列式

②变量代换与概率守恒

在生成模型中, 确保生成的概率分布合法(即概率密度函数的积分为 1)是核心约束。

根据概率论中的变量代换定理, 当变量的取值空间发生体积变形时, 其概率密度必须进行反向补偿。如果变换使体积膨胀了 k 倍, 那么该区域内的概率密度就必须缩小为原来的 $1/k$, 以保持总概率质量守恒。

在实际计算中, 模型通过计算变换函数雅可比矩阵的行列式的对数绝对值, 来精确修正目标分布的对数似然概率, 这是流模型能够精确建模复杂数据分布的数学基础。

案例 2-3: 图像变换中的面积缩放

在人工智能图像处理或特征映射中, 矩阵变换会改变数据的分布空间。本案例通过定义一个单位正方形(代表原始数据分布), 观察不同矩阵对其形态的改变。

矩阵 A (放大): $A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, 计算 $\det A = 2 \times 2 - 0 = 4$ 。验证图形面积将变为原来的 4 倍。

矩阵 B (压扁/奇异): $B = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$, 计算 $\det B = 1 \times 0 - 1 \times 0 = 0$ 。验证图形面积将变为 0, 所有点被压缩到一条线上。

```
import numpy as np
import matplotlib.pyplot as plt
# 解决中文显示乱码问题
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
# 定义单位正方形的顶点
square = np.array([[0,1,1,0,0],[0,0,1,1,0]])

# 定义变换矩阵
A = np.array([[2,0],[0,2]])
B = np.array([[1,1],[0,0]])

# 计算行列式并执行线性变换 Y = M @ X
det_A,trans_A = np.linalg.det(A),A @ square
det_B,trans_B = np.linalg.det(B),B @ square

# 绘图对比
fig,axes = plt.subplots(1,2,figsize=(10,4))

# --- 左图:矩阵 A 的缩放效果 ---
axes[0].plot(square[0],square[1],'k- ',label='原始单位正方形 (面积=1)')
axes[0].fill(trans_A[0],trans_A[1],color='skyblue',alpha=0.5,label='变换后图形 (面积=4)')
axes[0].set_title(f"面积缩放效应 (detA={det_A:.0f})")
axes[0].legend(loc='upper left')
axes[0].axis('equal')
axes[0].grid(True,linestyle=':')

# --- 右图:矩阵 B 的维度坍缩效果 ---
axes[1].plot(square[0],square[1],'k- ',label='原始单位正方形 (面积=1)')
axes[1].plot(trans_B[0],trans_B[1],color='red',linewidth=3,label='坍缩后的直线 (面积=0)')
```