矩阵函数和矩阵微分

Jingbo Xia

202601 —— 人工智能202301/02

本章符号约定

符号约定

向量记法：列向量用小写粗体表示，如 $\mathbf{x}, \mathbf{y}, \mathbf{b}$

矩阵记法：矩阵用大写粗体表示，如 $\mathbf{A}, \mathbf{B}, \mathbf{X}$

梯度记法：$\nabla_{\mathbf{x}} f = \frac{\partial f}{\partial \mathbf{x}}$

雅可比记法：$\mathbf{J}_{\mathbf{f}}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$

向量化操作：$\mathrm{vec}(\mathbf{X})$ 表示将矩阵按列堆叠成向量

迹运算：$\mathrm{Tr}(\mathbf{A})$ 表示矩阵的迹

转置与逆：$\mathbf{A}^T$ 表示转置，$\mathbf{A}^{-1}$ 表示逆

章节说明

*为非考核章节

目录

Outline

矩阵函数 I

矩阵函数的定义

矩阵函数是一种特殊的数学函数，其输入或输出（或两者）涉及矩阵结构。在现代数学、物理学和工程学中，矩阵函数构成了线性代数、微分几何、优化理论以及机器学习等领域的核心工具。

定义

矩阵函数可以形式化地定义为：

$$f : \mathcal{V} \to \mathcal{W}$$

其中，定义域 $\mathcal{V}$ 和值域 $\mathcal{W}$ 中至少一个空间包含矩阵（或更一般地，张量）结构。

按此定义，常见的函数类型包括：

标量函数：输入为矩阵，输出为标量

矩阵函数：输入为矩阵，输出也为矩阵

向量函数：输入为矩阵，输出为向量（或反之）

矩阵函数 II

矩阵函数的定义

矩阵函数的重要性

统一数学框架：提供将标量函数概念推广到矩阵空间的系统方法
优化与机器学习：损失函数、正则化项、激活函数等常用表达形式
物理与工程建模：描述多变量系统的状态变化和相互作用
数值计算：为高效算法设计提供理论基础

矩阵函数 III

矩阵函数的定义

矩阵函数的连续性、可微性与解析性

与标量函数类似，矩阵函数也有连续、可微和解析等概念.

连续性：若对任意收敛到 $\mathbf{A}$ 的矩阵序列 $\{\mathbf{A}_k\}$，有 $f(\mathbf{A}_k) \to f(\mathbf{A})$，则称 f 在 $\mathbf{A}$ 处连续

可微性：存在线性算子 $\mathcal{L}$ 使得

$$f(\mathbf{A}+\mathbf{H}) = f(\mathbf{A}) + \mathcal{L}(\mathbf{H}) + o(\|\mathbf{H}\|)$$

其中 $\mathcal{L}$ 称为 f 在 $\mathbf{A}$ 处的导数

解析性：若 f 在 $\mathbf{A}$ 的某个邻域内可展开为收敛的矩阵幂级数，则称 f 在 $\mathbf{A}$ 处解析

矩阵函数 I

矩阵函数的分类

基于函数的定义域和值域，矩阵函数可系统分类如下表所示。这种分类直接决定了导数的结构和计算方法。

表 1: 矩阵函数的系统分类

函数类型	定义域 $\mathcal{V}$	值域 $\mathcal{W}$	典型示例
标量对向量	$\mathbb{R}^n$	$\mathbb{R}$	$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$
向量对向量	$\mathbb{R}^n$	$\mathbb{R}^m$	$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$
标量对矩阵	$\mathbb{R}^{m\times n}$	$\mathbb{R}$	$f(\mathbf{X}) = \mathrm{tr}(\mathbf{X}^T\mathbf{X})$
向量对矩阵	$\mathbb{R}^{m\times n}$	$\mathbb{R}^p$	$\mathbf{f}(\mathbf{X}) = \mathrm{vec}(\mathbf{X})$
矩阵对矩阵	$\mathbb{R}^{m\times n}$	$\mathbb{R}^{p\times q}$	$f(\mathbf{X}) = \mathbf{X}^T\mathbf{X}$

矩阵函数 II

矩阵函数的分类

按函数形式分类

除了按输入输出类型分类，矩阵函数还可按其具体数学形式分类：

- **线性函数**：$f(\mathbf{X}) = \mathbf{A}\mathbf{X} + \mathbf{B}$ 或 $f(\mathbf{X}) = \mathrm{tr}(\mathbf{A}^T\mathbf{X})$
- **二次型函数**：$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ 或 $f(\mathbf{X}) = \mathrm{tr}(\mathbf{X}^T\mathbf{A}\mathbf{X})$
- **复合函数**：由基本函数通过加、减、乘、除、复合等运算构成
- **特殊矩阵函数**：行列式、迹、秩、谱函数等

每种形式都有其独特的性质和求导规则，掌握这些基本形式是学习矩阵微分的关键。

矩阵函数 I

*例：矩阵指数函数和有向无环图(DAG)的判别

指数函数

指数函数 e^x 在 $x=0$处的泰勒级数（麦克劳林级数）展开为：

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots .$$

推导： 因为 $\frac{d}{dx}e^x = e^x$，且在 $x=0$ 处各阶导数都等于 1：$f^{(n)}(0) = 1 \quad (n = 0, 1, 2, \dots)$
代入泰勒公式 $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n$ 即得上述展开。

收敛性： 该级数对任意实数 x 绝对收敛，收敛半径 $R = \infty$。

特殊值举例： $e = e^1 = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \cdots \approx 2.71828$

矩阵函数 II

*例：矩阵指数函数和有向无环图(DAG)的判别

矩阵的指数函数

设 $W \in \mathbb{R}^{d \times d}$ 是一个方阵，其矩阵指数 e^W 定义为以下收敛的幂级数：

$$e^W = \sum_{k=0}^{\infty} \frac{W^k}{k!} = I + W + \frac{W^2}{2!} + \frac{W^3}{3!} + \cdots$$

其中 I 是 $d \times d$ 单位矩阵，$W^0 = I$。

收敛性：此级数对任意方阵均收敛。

矩阵函数 III

*例：矩阵指数函数和有向无环图(DAG)的判别

有向无环图(DAG)的判别定理

对于 d 个顶点的有向图，设其邻接矩阵为 $W = [w_{ij}]_{d \times d}$，其中 $w_{ij} \in \{0, 1\}$ 表示从顶点 i 到顶点 j 是否存在有向边。那么：

$$\text{该有向图是无环图（DAG）} \iff \operatorname{Tr}(e^W) = d$$

即矩阵指数的迹等于顶点个数时，图不含任何有向环；反之，若迹大于 d，则图中至少存在一个有向环。

矩阵函数 IV

***例：矩阵指数函数和有向无环图(DAG)的判别**

【图论】——《离散数学》

在图论中，对于无权有向图（$w_{ij} \in \{0,1\}$），邻接矩阵的 k 次幂 W^k 的每个元素具有明确的组合意义：

$$(W^k)_{ij} = \text{从顶点 } i \text{ 到顶点 } j \text{ 的长度为 } k \text{ 的（不同）有向路径的数量}$$

特别地，对角线元素：

$$(W^k)_{ii} = \text{从顶点 } i \text{ 出发并回到自身的、长度为 } k \text{ 的有向环的数量}$$

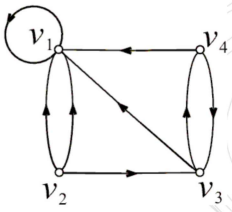
矩阵函数 V

***例：矩阵指数函数和有向无环图(DAG)的判别**

【图论】——《离散数学》

Graph、邻接矩阵、通路和回路

例 有向图D如图所示，求 A, A^2, A^3, A^4，并回答诸问题：
(1) D 中长度为1, 2, 3, 4的通路各有多少条？其中回路分别为多少条？
(2) D 中长度小于或等于4的通路为多少条？其中有多少条回路？



$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad A^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{bmatrix} \quad A^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 1 \\ 4 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 \end{bmatrix}$$

(1) D中长度为1的通路为8条，其中有1条是回路.
D中长度为2的通路为11条，其中有3条是回路.
D中长度为3和4的通路分别为14和17条，回路分别为1与3条.
(2) D中长度小于等于4的通路为50条，其中有8条是回路.

矩阵函数 VI

***例：矩阵指数函数和有向无环图(DAG)的判别**

证明“有向无环图(DAG)的判别定理”

对于 d 个顶点的有向图，设其邻接矩阵为 $W = [w_{ij}]_{d\times d}$，其中 $w_{ij} \in \{0,1\}$ 表示从顶点 i 到顶点 j 是否存在有向边。那么：

$$\text{该有向图是无环图（DAG）} \iff \mathrm{Tr}(e^W) = d$$

证明：（i）首先我们做一些运算准备。我们将矩阵指数的定义代入迹运算，有

$$\begin{aligned} \mathrm{Tr}(e^W) &= \mathrm{Tr}\left(\sum_{k=0}^{\infty} \frac{W^k}{k!}\right) \\ &= \sum_{k=0}^{\infty} \frac{1}{k!} \mathrm{Tr}(W^k) \quad \text{（利用迹的线性性）} \end{aligned}$$

由于 $W^0 = I$，且 $\mathrm{Tr}(I) = d$，上式可写为：

$$\mathrm{Tr}(e^W) = d + \sum_{k=1}^{\infty} \frac{\mathrm{Tr}(W^k)}{k!} \tag{1}$$

由【图论】中结论，$\mathrm{Tr}(W^k)$ 是图中所有长度为 k 的有向环的总数：

$$\mathrm{Tr}(W^k) = \sum_{i=1}^{d} (W^k)_{ii} = \text{全体长度为 } k \text{ 的有向环的数量}$$

同时，又由于邻接矩阵元素 $w_{ij} \geq 0$，所有项 $(W^k)_{ii}$ 均为非负整数，因此有：

$$\mathrm{Tr}(W^k) \geq 0 \quad \text{对所有 } k \geq 1 \text{ 成立} \tag{2}$$

（ii）下面我们证明“**DAG => trace = d**”。
若图是无环图（DAG），则图中不存在任何有向环。因此，对任意 $k \geq 1$，都有 $\mathrm{Tr}(W^k) = 0$。将其代入展开式(1)，立即得到：

$$\mathrm{Tr}(e^W) = d + 0 = d$$

(iii) 最后我们证明“trace = d =>DAG”。
若已知 $\mathrm{Tr}(e^W) = d$，观察其展开式 $d + \sum_{k=1}^{\infty} \frac{\mathrm{Tr}(W^k)}{k!}$。由（2）知，该级数每一项 $\frac{\mathrm{Tr}(W^k)}{k!}$ 都非负，要使总和等于 d，必须满足：

$$\mathrm{Tr}(W^k) = 0 \quad \text{对所有 } k \geq 1$$

这意味着图中不存在任何长度的有向环，因此该图必然是一个有向无环图（DAG）。

矩阵函数 I

***在机器学习与因果发现中的应用：Structure learning**

在基于梯度的结构学习（如 NOTEARS 方法）中，该结论被巧妙地推广为连续优化问题的约束：

将离散的邻接矩阵 W 松弛为连续的权重矩阵。

为避免正负抵消问题，使用元素平方 $W \odot W$（哈达玛积）确保非负性。

构造一个可微的无环约束函数：

$$h(W) = \operatorname{Tr}\left(e^{W \odot W}\right) - d = 0$$

在优化中，通过梯度方法最小化损失函数的同时，推动 $h(W) \to 0$。

此时的 W 元素是实数，但 $W \odot W$ 非负，上述约束是原结论在连续优化背景下的一种有效且实用的松弛。

这一推广使得从数据中学习大规模有向无环图结构的连续优化成为可能，避免了复杂的组合搜索。

矩阵函数 II

***在机器学习与因果发现中的应用：Structure learning**

1. 扩展阅读——矩阵函数在建模中的典型应用——以一篇因果推断论文为例。（矩阵函数用以统计通路数量——请联系《离散数学》图论部分相关结论）

Theorem 1. *A matrix $W \in \mathbb{R}^{d\times d}$ is a DAG if and only if*

$$h(W) = \operatorname{tr}\left(e^{W\circ W}\right) - d = 0, \tag{5}$$

where $\circ$ is the Hadamard product and e^A is the matrix exponential of A. Moreover, $h(W)$ has a simple gradient

$$\nabla h(W) = \left(e^{W\circ W}\right)^T \circ 2W, \tag{6}$$

and satisfies all of the desiderata (a)-(d).

Zheng, Xun, et al. "DAGs with no tears: Continuous optimization for structure learning." *Advances in neural information processing systems* 31 (2018). (Link)

https://arxiv.org/abs/1803.01422

矩阵函数 III

***在机器学习与因果发现中的应用：Structure learning**

论文原文：Structure learning的定义

Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ be a data matrix consisting of n i.i.d. observations of the random vector $X = (X_1, ..., X_d)$ and let $\mathbb{D}$ denote the (discrete) space of DAGs $G = (V, E)$ on d nodes. Given $\boldsymbol{X}$, we seek to learn a DAG $G \in \mathbb{D}$ (also called a Bayesian network) for the joint distribution $P(X)$.

矩阵函数 IV

***在机器学习与因果发现中的应用：Structure learning**

论文原文：Structure learning的定义

Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a data matrix consisting of n i.i.d. observations of the random vector $X = (X_1, ..., X_d)$ and let $\mathbb{D}$ denote the (discrete) space of DAGs $G = (V, E)$ on d nodes.
Given $\boldsymbol{X}$, we seek to learn a DAG $G \in \mathbb{D}$ (also called a Bayesian network) for the joint distribution $P(X)$.

论文原文：邻接矩阵的记号准备

Any $W \in \mathbb{R}^{d \times d}$ defines a graph on d nodes in the following way: Let $A(W) \in \{0, 1\}^{d \times d}$ be the binary matrix such that $[A(W)]_{ij} = 1 <=> w_{ij} \neq= 0$ and zero otherwise; then $A(W)$ defines the adjacency matrix of a directed graph $G(W)$.

矩阵函数 V

***在机器学习与因果发现中的应用：Structure learning**

论文原文：Structure equation model（SEM），和 Loss函数的定义

$W = [w_1||w_d]$ defines a linear SEM by $X_j = w_j^T X + z_j$, where $X = (X_1, ..., X_d)$ is a random vector, and $z = (z_1, ..., z_d)$ is a random noise vector.

In this paper, we focus on linear SEM and the least-squares (LS) loss

$$l(W; \mathbf{X}) = \frac{1}{2n} ||\mathbf{X} - \mathbf{X}W||_F^2.$$

矩阵函数 VI

***在机器学习与因果发现中的应用：Structure learning**

论文原文：传统的Causal Discovery优化问题的表述

$$\min_{W \in \mathbb{R}^{d \times d}} F(W)$$
$$\text{subject to: } G(W) \in \mathbb{D}$$

其中

$$F(W) = \ell(W; \mathbf{X}) + \lambda ||W||_1 = \frac{1}{2n} ||\mathbf{X} - \mathbf{X}W||_F^2 + \lambda ||W||_1$$

矩阵函数 VII

***在机器学习与因果发现中的应用：Structure learning**

论文原文：传统的Structure Learning优化问题的表述

$$\min_{W\in\mathbb{R}^{d\times d}} F(W)$$
$$\text{subject to: } G(W)\in\mathbb{D}$$

其中

$$F(W)=\ell(W;\mathbf{X})+\lambda||W||_1=\frac{1}{2n}||\mathbf{X}-\mathbf{X}W||_F^2+\lambda||W||_1$$

简述

$$\min_{W\in\mathbb{R}^{d\times d}} F(W)=\tfrac{1}{2n}||\mathbf{X}-\mathbf{X}W||_F^2+\lambda||W||_1$$
$$\text{subject to: } W \text{ is a DAG}$$

矩阵函数 VIII

***在机器学习与因果发现中的应用：Structure learning**

传统的Structure learning算法模型

$$\min_{W\in\mathbb{R}^{d\times d}} F(W)=\tfrac{1}{2n}||\mathbf{X}-\mathbf{X}W||_F^2+\lambda||W||_1$$
$$\text{subject to: } W \text{ is a DAG}$$

vs.

NOTEARS算法中的方法改良

$$\min_{W\in\mathbb{R}^{d\times d}} F(W)=\tfrac{1}{2n}||\mathbf{X}-\mathbf{X}W||_F^2+\lambda||W||_1$$
$$\text{subject to: } h(W)=\operatorname{Tr}\left(e^{W\odot W}\right)-d=0$$

NOTEARS一文的方法突破体现在哪里？为什么？

Outline

矩阵微分 I

标量函数对向量求导（梯度）-定义和几何理解

矩阵微分根据函数输出与输入变量的类型不同，导数具有不同的结构与维度。以下均采用**分母布局**。

设 $f:\mathbb{R}^n\to\mathbb{R}$ 是标量函数，输入为列向量 $\boldsymbol{x}=[x_1,x_2,\ldots,x_n]^T\in\mathbb{R}^n$。

定义：梯度向量

$$\frac{\partial f}{\partial \boldsymbol{x}}=\nabla_{\boldsymbol{x}} f=\begin{bmatrix}\frac{\partial f}{\partial x_1}\\ \frac{\partial f}{\partial x_2}\\ \vdots\\ \frac{\partial f}{\partial x_n}\end{bmatrix}\in\mathbb{R}^{n\times 1}$$

矩阵微分 II

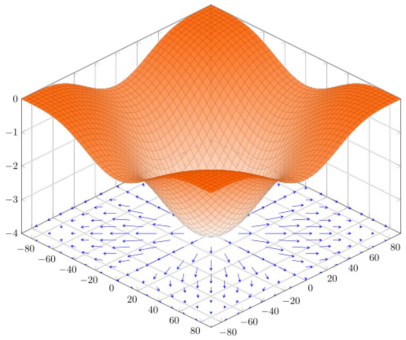
标量函数对向量求导（梯度）-定义和几何理解

(1) **梯度的几何学理解（参考Wikipedia）**：想象你站在三维曲面的某一点上，这个曲面由函数$z=f(x,y)$描述。梯度∇f在x-y平面上的投影，恰好指向使函数值z增长最快的水平方向。如果你沿着梯度方向前进，就好像沿着山坡最陡峭的方向向上爬升。在多维空间中，这一直观依然成立：梯度指向了函数值增长最快的方向。
模长意义：梯度的模长$\|\nabla f\|$量化了这个"最快"的增长速率。模长越大，意味着函数在该点附近变化越剧烈；模长为零（梯度为零向量）则对应函数的驻点（可能是极值点或鞍点）。
与等高线的关系：在二维情形下，函数的等高线是平面上使函数值相等的曲线。梯度在任意点处总是垂直于通过该点的等高线，并指向函数值增加的一侧。这一性质推广到高维：梯度垂直于函数的等值面。

矩阵微分 III

标量函数对向量求导（梯度）-定义和几何理解

图例： Consider a surface whose height above sea level at point (x, y) is $H(x, y)$. The gradient of H at a point is a vector pointing in the direction of the steepest slope or grade at that point. The steepness of the slope at that point is given by the magnitude of the gradient vector.

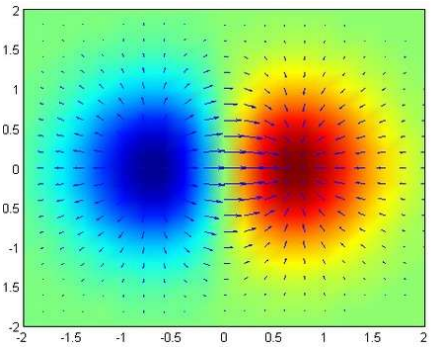


$f(x, y) = (\cos^2 x + \cos^2 y)^2$

矩阵微分 IV

标量函数对向量求导（梯度）-定义和几何理解

(2) **梯度的热力学理解（参考Wikipedia）：** Consider a room in which the temperature is given by a scalar field, T, so at each point (x, y, z) the temperature is $T(x, y, z)$. At each point in the room, the gradient of T at that point will show the direction in which the temperature rises most quickly. The magnitude of the gradient will determine how fast the temperature rises in that direction.



$f(x, y) = xe^{(x2+y2)}$

矩阵微分 I

标量函数对向量求导（梯度）-梯度和方向导数

方向导数(Directional Derivative)的定义
设函数 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}_0 \in \mathbb{R}^d$ 的某个邻域内有定义。$\mathbf{u}$ 是一个单位方向向量 ($\|\mathbf{u}\|_2 = 1$)。若极限

$$D_{\mathbf{u}} f(\boldsymbol{x}_0) = \lim_{h \to 0} \frac{f(\boldsymbol{x}_0 + h\mathbf{u}) - f(\boldsymbol{x}_0)}{h}$$

存在，则称此极限值为函数 f 在点 $\boldsymbol{x}_0$ 处沿方向 $\mathbf{u}$ 的方向导数 [a]。

[a]注：方向导数是一个标量，表示函数在某个方向上的变化率。

矩阵微分 II

标量函数对向量求导（梯度）-梯度和方向导数

梯度与方向导数的关系

若函数 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}$ 处可微，则函数在该点处沿任意单位向量 $\mathbf{u}$ 的方向导数必存在，且可以表示为梯度向量 $\nabla f(\boldsymbol{x})$ 与 $\mathbf{u}$ 的内积：

$$D_{\mathbf{u}} f(\boldsymbol{x}) = (\nabla f(\boldsymbol{x}), \mathbf{u}) = \nabla f(\boldsymbol{x})^T \mathbf{u}$$

分析：设θ 是梯度 $\nabla f(\boldsymbol{x})$ 与方向向量 $\mathbf{u}$ 之间的夹角，由向量内积性质，有

$$D_{\mathbf{u}} f(\boldsymbol{x}) = \|\nabla f(\boldsymbol{x})\|_2 \cdot \|\mathbf{u}\|_2 \cdot \cos\theta = \|\nabla f(\boldsymbol{x})\|_2 \cos\theta.$$

（证明过程略，使用矩阵函数的多项式展开）

矩阵微分 III

标量函数对向量求导（梯度）-梯度和方向导数

$$D_{\mathbf{u}} f(\boldsymbol{x}) = \|\nabla f(\boldsymbol{x})\|_2 \cdot \|\mathbf{u}\|_2 \cdot \cos\theta = \|\nabla f(\boldsymbol{x})\|_2 \cos\theta.$$

总结可知：
最速上升方向：当 $\theta = 0$ 时，$\mathbf{u}$ 与 $\nabla f(\boldsymbol{x})$ 同向，$D_{\mathbf{u}} f(\boldsymbol{x})$ 取得最大值 $\|\nabla f(\boldsymbol{x})\|_2$。
最速下降方向：当 $\theta = \pi$ 时，$\mathbf{u}$ 与 $\nabla f(\boldsymbol{x})$ 反向，$D_{\mathbf{u}} f(\boldsymbol{x})$ 取得最小值 $-\|\nabla f(\boldsymbol{x})\|_2$。
变化率为零方向：当 $\theta = \frac{\pi}{2}$ 时，$\mathbf{u} \perp \nabla f(\boldsymbol{x})$，$D_{\mathbf{u}} f(\boldsymbol{x}) = 0$。
梯度是一个向量，它综合了所有方向导数的信息，并指引了函数值增长最快的几何方位。

矩阵微分 I

标量函数对向量求导（梯度）-计算示例

一般而言，对矩阵函数$f(\boldsymbol{x})$求梯度计算$\nabla f(\boldsymbol{x})$ 可以掌握两个技巧。
一是按$\boldsymbol{x}$的分量进行相应计算，从头推理。
二是记忆一些常见结论，并与传统意义下$f(x)$的微分计算做结果联想。

例

当$f(\boldsymbol{x}) := \boldsymbol{x}^T \boldsymbol{x} = x_1^2 + \cdots + x_n^2$, 我们有

$$\nabla f(\boldsymbol{x}) = 2\boldsymbol{x}.$$

矩阵微分 II

标量函数对向量求导（梯度）-计算示例

计算示例：

例1： 若$f(\boldsymbol{x}) = x_1^2 + 3x_2 + \sin(x_3)$，则

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 3 \\ \cos(x_3) \end{bmatrix}$$

在点$\boldsymbol{x} = (1, 2, \pi/2)$处，$\nabla f = [2, 3, 0]^T$，表示在该点处，沿$x_1$方向增长最快（斜率2），沿$x_2$方向次之（斜率3），沿$x_3$方向瞬时变化率为0。

矩阵微分 III

标量函数对向量求导（梯度）-计算示例

例2（二元二次函数）： 考虑$f(x, y) = x^2 + 2y^2 - 4x + 2y + 5$，其梯度为

$$\nabla f = ?$$

矩阵微分 IV

标量函数对向量求导（梯度）-计算示例

例2（二元二次函数）： 考虑$f(x, y) = x^2 + 2y^2 - 4x + 2y + 5$，其梯度为

$$\nabla f = \begin{bmatrix} 2x - 4 \\ 4y + 2 \end{bmatrix}$$

在点$(2, -0.5)$处，$\nabla f = [0, 0]^T$，这是函数的驻点（实际为极小值点）。

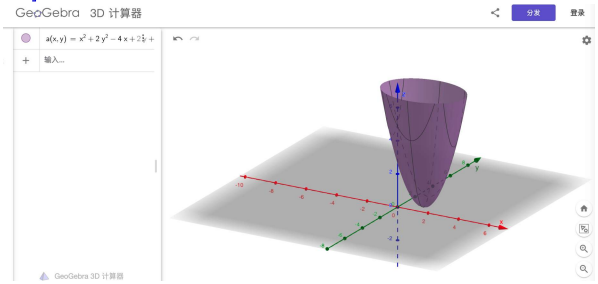
矩阵微分 V

标量函数对向量求导（梯度）-计算示例

例2（二元二次函数）：考虑$f(x,y)=x^2+2y^2-4x+2y+5$，其梯度为

$$\nabla f=\begin{bmatrix}2x-4\\4y+2\end{bmatrix}$$

在点$(2,-0.5)$处，$\nabla f=[0,0]^T$，这是函数的驻点（实际为极小值点）。



矩阵微分 VI

标量函数对向量求导（梯度）-计算示例

例3（多元线性函数）：$f(\boldsymbol{x})=\boldsymbol{a}^T\boldsymbol{x}=(\boldsymbol{a},\boldsymbol{x})=a_1x_1+a_2x_2+\cdots+a_nx_n$，其梯度为常数向量

$$\nabla f=?$$

矩阵微分 VII

标量函数对向量求导（梯度）-计算示例

例3（多元线性函数）：$f(\boldsymbol{x})=\boldsymbol{a}^T\boldsymbol{x}=(\boldsymbol{a},\boldsymbol{x})=a_1x_1+a_2x_2+\cdots+a_nx_n$，其梯度为常数向量

$$\nabla f=\begin{bmatrix}a_1\\a_2\\\vdots\\a_n\end{bmatrix}=\boldsymbol{a}$$

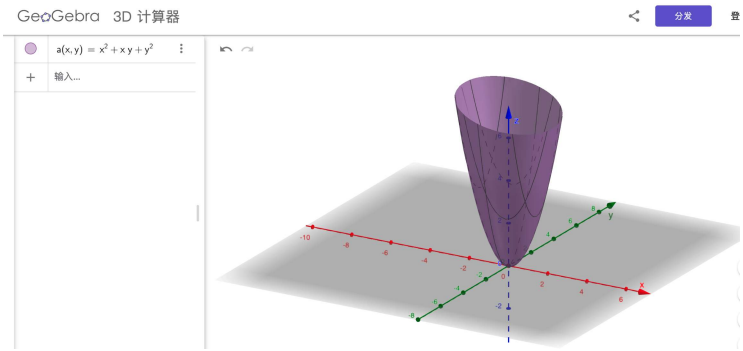
这表明线性函数的梯度处处相同，增长最快的方向始终是系数向量$\boldsymbol{a}$的方向。

矩阵微分 VIII

标量函数对向量求导（梯度）-计算示例

例4（二元二次函数，椭圆抛物面）： $f(x,y)=x^2+xy+y^2$，梯度为

$$\nabla f = ?$$



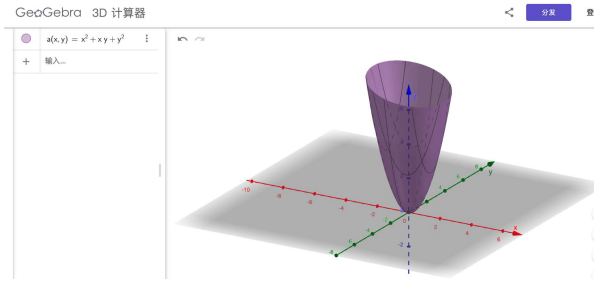
矩阵微分 IX

标量函数对向量求导（梯度）-计算示例

例4（二元二次函数，椭圆抛物面）： $f(x,y)=x^2+xy+y^2$，梯度为

$$\nabla f = \begin{bmatrix} 2x+y \\ x+2y \end{bmatrix}$$

在点$(1,1)$处，$\nabla f=[3,3]^T$，指向第一象限45度方向，模长为$3\sqrt{2}$。

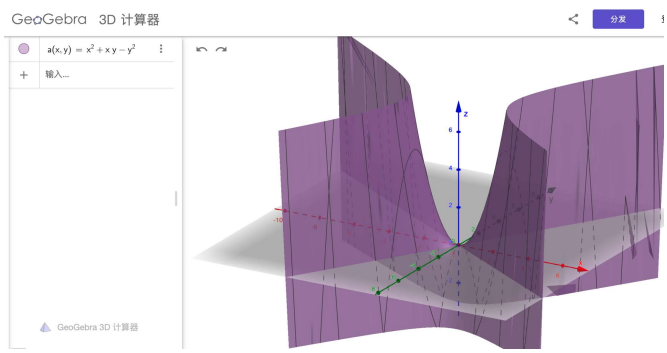


矩阵微分 X

标量函数对向量求导（梯度）-计算示例

例5（二元二次函数,双曲抛物面）： $f(x,y)=x^2+xy-y^2$，梯度为

$$\nabla f = \emptyset$$



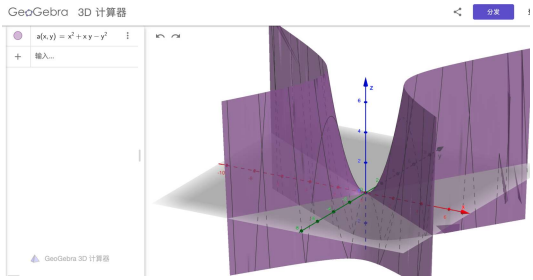
矩阵微分 XI

标量函数对向量求导（梯度）-计算示例

例5（二元二次函数,双曲抛物面）： $f(x,y)=x^2+xy-y^2$，梯度为

$$\nabla f=\begin{bmatrix}2x+y\\ x-2y\end{bmatrix}$$

在点$(1,1)$处，$\nabla f=[3,-1]^T$，指向第四象限约$-18.4°$方向，模长为$\sqrt{10}$。



矩阵微分 I

标量函数对矩阵求导

设 $f:\mathbb{R}^{m\times n}\to\mathbb{R}$，输入为矩阵 $\boldsymbol{X}=[X_{ij}]\in\mathbb{R}^{m\times n}$，输出为标量 $f(\boldsymbol{X})$。

定义：矩阵梯度

$$\frac{\partial f}{\partial \boldsymbol{X}}=\begin{bmatrix}\frac{\partial f}{\partial X_{11}} & \frac{\partial f}{\partial X_{12}} & \cdots & \frac{\partial f}{\partial X_{1n}}\\ \frac{\partial f}{\partial X_{21}} & \frac{\partial f}{\partial X_{22}} & \cdots & \frac{\partial f}{\partial X_{2n}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f}{\partial X_{m1}} & \frac{\partial f}{\partial X_{m2}} & \cdots & \frac{\partial f}{\partial X_{mn}}\end{bmatrix}\in\mathbb{R}^{m\times n}$$

矩阵微分 II

标量函数对矩阵求导

解释与性质

(1) **直观理解**：将矩阵 $\boldsymbol{X}$ 视为 mn 个独立变量，导数将这些偏导数按原矩阵形状排列。

(2) **向量化方法**：可先将矩阵向量化 $\text{vec}(\boldsymbol{X}) \in \mathbb{R}^{mn}$，计算梯度 $\frac{\partial f}{\partial \text{vec}(\boldsymbol{X})}$，再reshape回 $m \times n$。

(3) **微分形式**：常用方法是先求全微分 df，利用迹技巧：

$$df = \text{tr}\left(\boldsymbol{A}^T d\boldsymbol{X}\right) \quad \Longrightarrow \quad \frac{\partial f}{\partial \boldsymbol{X}} = \boldsymbol{A}$$

这是因为 $\text{tr}(\boldsymbol{A}^T d\boldsymbol{X}) = \sum_{i,j} A_{ij} dX_{ij}$。

矩阵微分 I

*向量函数对向量求导（雅可比矩阵）

设 $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$，输入 $\boldsymbol{x} \in \mathbb{R}^n$，输出 $\mathbf{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]^T \in \mathbb{R}^m$。

定义：雅可比矩阵

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

矩阵微分 II

***向量函数对向量求导（雅可比矩阵）**

解释与性质

(1) **矩阵结构：**

行数 = 输出维度 m，每行对应一个输出分量 f_i 的梯度（转置）

列数 = 输入维度 n，每列对应所有输出分量对 x_j 的偏导

(2) **链式法则：**设 $\boldsymbol{y} = \mathbf{g}(\boldsymbol{x})$，$\boldsymbol{z} = \mathbf{f}(\boldsymbol{y})$，则

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}} \cdot \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$$

其中乘法为矩阵乘法，维度：$(p \times m) \cdot (m \times n) = p \times n$。

(3) **重要特例：**

当 $m = 1$ 时，退化为梯度向量的转置（行向量）

当 $\mathbf{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$（线性变换）时，$\frac{\partial \mathbf{f}}{\partial \boldsymbol{x}} = \boldsymbol{A}$

矩阵微分 I

***向量函数对矩阵求导（高阶张量）**

设 $\mathbf{f} : \mathbb{R}^{m \times n} \to \mathbb{R}^p$，输入为矩阵 $\boldsymbol{X} \in \mathbb{R}^{m \times n}$，输出为向量 $\mathbf{f}(\boldsymbol{X}) \in \mathbb{R}^p$。

定义：三维张量

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{X}} \in \mathbb{R}^{p \times m \times n}$$

其中第 k 个切片（$k = 1, \ldots, p$）是 $\frac{\partial f_k}{\partial \boldsymbol{X}} \in \mathbb{R}^{m \times n}$。

解释与性质

(1) **张量结构**：可看作 p 个 $m \times n$ 矩阵堆叠而成，每个对应一个输出分量的矩阵导数。

(2) **实际处理**：通常避免直接操作三维张量，而是：

将矩阵向量化：$\frac{\partial \mathbf{f}}{\partial \mathrm{vec}(\boldsymbol{X})} \in \mathbb{R}^{p \times (mn)}$

逐个分量求导：对每个 f_k，计算 $\frac{\partial f_k}{\partial \boldsymbol{X}} \in \mathbb{R}^{m \times n}$

(3) **常见场景**：神经网络中损失函数对权重矩阵的求导属于此类。

Outline

常用矩阵微分计算 I

二次型的梯度计算

例：

计算二次型 $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ 对向量 $\mathbf{x}$ 的梯度 $\frac{\partial f}{\partial \mathbf{x}}$。其中，$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ 为列向量，$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ 为任意 n 阶方阵（不要求对称）。

常用矩阵微分计算 II

二次型的梯度计算

例：

计算二次型 $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ 对向量 $\mathbf{x}$ 的梯度 $\frac{\partial f}{\partial \mathbf{x}}$。其中，$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ 为列向量，$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ 为任意 n 阶方阵（不要求对称）。

1. 如何将函数 $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ 明确地展开为求和形式，以便进行分量求导？
2. 对其中任意一个分量 x_k 求偏导 $\frac{\partial f}{\partial x_k}$ 时，如何处理双重求和？
3. 当矩阵 $\mathbf{A}$ 满足对称性（即 $\mathbf{A}^T = \mathbf{A}$）时，梯度公式可以简化为什么形式？这个特例在哪些实际应用中常见？

常用矩阵微分计算 III

二次型的梯度计算

解：

1. 展开为求和形式 将二次型函数展开为明确的双重求和，是进行分量求导的基础：

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j.$$

2. 对分量 x_k 求偏导 考虑双重求和 $\sum_i \sum_j a_{ij}x_i x_j$ 中，包含变量 x_k 的项出现在两种情况下：

   情况一： 当第一个下标 $i = k$ 时，项为 $a_{kj}x_k x_j$。将 x_j 视为常数，该项对 x_k 的偏导为 $a_{kj}x_j$。

   情况二： 当第二个下标 $j = k$ 时，项为 $a_{ik}x_i x_k$。将 x_i 视为常数，该项对 x_k 的偏导为 $a_{ik}x_i$。

常用矩阵微分计算 IV

二次型的梯度计算

因此，x_k 的偏导数是这两种情况贡献之和：

$$\frac{\partial f}{\partial x_k} = \sum_{j=1}^{n} a_{kj}x_j + \sum_{i=1}^{n} a_{ik}x_i.$$

3. 转换为矩阵形式 观察上式的两项：

   第一项 $\sum_{j=1}^{n} a_{kj}x_j$ 是矩阵 $\mathbf{A}$ 的第 k 行与向量 $\mathbf{x}$ 的内积，即矩阵乘法 $\mathbf{Ax}$ 的第 k 个分量 $(\mathbf{Ax})_k$。

   第二项 $\sum_{i=1}^{n} a_{ik}x_i$ 是矩阵 $\mathbf{A}^T$ 的第 k 行与向量 $\mathbf{x}$ 的内积，即矩阵乘法 $\mathbf{A}^T\mathbf{x}$ 的第 k 个分量 $(\mathbf{A}^T\mathbf{x})_k$。

常用矩阵微分计算 V

二次型的梯度计算

所以，

$$\frac{\partial f}{\partial x_k} = (\mathbf{Ax})_k + (\mathbf{A}^T\mathbf{x})_k = ((\mathbf{A} + \mathbf{A}^T)\mathbf{x})_k.$$

4. 得到梯度向量 由于上述关系对每一个分量 $k = 1, 2, \ldots, n$ 都成立，将所有偏导数组合成列向量，即得到最终的梯度表达式：

$$\boxed{\frac{\partial f}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}}.$$

B站讲义：https://www.bilibili.com/video/BV1Du4y137Yv?t=19.2

常用矩阵微分计算 VI

二次型的梯度计算

对称矩阵的特例

当矩阵 $\mathbf{A}$ 对称（$\mathbf{A}^T = \mathbf{A}$）时，上述公式简化为：

$$\frac{\partial(\mathbf{x}^T\mathbf{Ax})}{\partial \mathbf{x}} = 2\mathbf{Ax}.$$

常用矩阵微分计算 VII

二次型的梯度计算

常见应用

这个特例在优化和机器学习中极为常见，例如：

在最小二乘问题中，目标函数 $\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2$ 的二次项部分。

在支持向量机、岭回归等模型的损失函数中，正则化项通常为 $\mathbf{x}^T\mathbf{x}$（此时 $\mathbf{A}=\mathbf{I}$）。

在牛顿法等优化算法中，需要计算海森矩阵（Hessian Matrix），对于二次型函数，其海森矩阵即为 $2\mathbf{A}$（或 $\mathbf{A}+\mathbf{A}^T$）。

掌握这一基础公式的推导，是理解和计算更复杂矩阵微分的关键。

常用矩阵微分计算 VIII

二次型的梯度计算

回忆线性回归的Loss函数：

$$\mathcal{J}(\vec{w})=\frac{1}{n}\sum_{i=1}^{n}(y_i-f(\vec{x}_i,\vec{w}))^2=\frac{1}{n}(\vec{y}-X\vec{w})^T(\vec{y}-X\vec{w}):=\frac{1}{n}\|\vec{y}-X\vec{w}\|_2^2.$$

对$\mathcal{J}(\vec{w})$计算梯度:

$$\begin{aligned}\frac{\partial\mathcal{J}(\vec{w})}{\partial\vec{w}}&=\frac{\partial\left(\frac{1}{n}(\vec{y}-X\vec{w})^T(\vec{y}-X\vec{w})\right)}{\partial\vec{w}}\\&=\frac{1}{n}\frac{\partial\left((\vec{y}^T-\vec{w}^TX^T)(\vec{y}-X\vec{w})\right)}{\partial\vec{w}}\\&=\frac{1}{n}\frac{\partial(\vec{y}^T\vec{y}-\vec{y}^TX\vec{w}-\vec{w}^TX^T\vec{y}+\vec{w}^TX^TX\vec{w})}{\partial\vec{w}}\\&=?\end{aligned}$$

常用矩阵微分计算 IX

二次型的梯度计算

对$\mathcal{J}(\vec{w})$计算梯度:

$$\begin{aligned}\frac{\partial\mathcal{J}(\vec{w})}{\partial\vec{w}}&=\frac{\partial\left(\frac{1}{n}(\vec{y}-X\vec{w})^T(\vec{y}-X\vec{w})\right)}{\partial\vec{w}}\\&=\frac{1}{n}\frac{\partial\left((\vec{y}^T-\vec{w}^TX^T)(\vec{y}-X\vec{w})\right)}{\partial\vec{w}}\\&=\frac{1}{n}\frac{\partial(\vec{y}^T\vec{y}-\vec{y}^TX\vec{w}-\vec{w}^TX^T\vec{y}+\vec{w}^TX^TX\vec{w})}{\partial\vec{w}}\\&=\frac{1}{n}(0-X^T\vec{y}-X^T\vec{y}+2X^TX\vec{w})\\&=\frac{2}{n}(-X^T\vec{y}+X^TX\vec{w})\end{aligned}$$

常用矩阵微分计算 X

二次型的梯度计算

结果的一般形式

设 $f(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})^T \boldsymbol{C}(\boldsymbol{D}\boldsymbol{x} + \boldsymbol{e})$，其中 $\boldsymbol{A} \in \mathbb{R}^{m \times n}$，$\boldsymbol{C} \in \mathbb{R}^{m \times p}$，$\boldsymbol{D} \in \mathbb{R}^{p \times n}$。

$$\frac{\partial f}{\partial \boldsymbol{x}} = \boldsymbol{A}^T \boldsymbol{C}(\boldsymbol{D}\boldsymbol{x} + \boldsymbol{e}) + \boldsymbol{D}^T \boldsymbol{C}^T (\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$$

常用矩阵微分计算 XI

二次型的梯度计算

你会计算LASSO回归的目标函数的梯度了吗？

矩阵微分计算 I
迹的微分计算——常见结论

定理 (矩阵的Frobenius norm和矩阵的迹)

对于任意$A \in \mathbb{R}^{m\times n}$, 有

$$||A||_F^2 = Tr(A^T A).$$

证明：直接验算可得。

矩阵微分计算 II
迹的微分计算——常见结论

定理 (Trace(XA)对X的微分)

对于$X \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{n\times m}$, 我们有

$$\frac{\partial Tr(XA)}{\partial X} = A^T.$$

证明：直接验算可得。

矩阵微分计算 III
迹的微分计算——常见结论

定理 (推论)

设$X \in \mathbb{R}^{m\times n}$，则对于任意$A \in \mathbb{R}^{m\times n}$，我们有

$$\frac{\partial Tr(X^T A)}{\partial X} = \frac{\partial Tr(A^T X)}{\partial X} = A. \tag{3}$$

证明：由前定理，我们知 $\frac{\partial Tr(X^T A)}{\partial X^T} = A^T.$
结果取转置，有 $\frac{\partial Tr(X^T A)}{\partial X} = A.$

又因为$Tr(X^T A) = Tr((X^T A)^T) = Tr(A^T X)$，所以$\frac{\partial Tr(A^T X)}{\partial X} = A$。

习题：

设$X, A \in \mathbb{R}^{n\times n}$, 请证明

$$\frac{\partial Tr(XAX^T)}{\partial X} = XA^T + XA.$$

提示：
(1)使用链式法则；
(2) 对于任意矩阵 $C \in \mathbb{R}^{m\times n}$ 和 $D \in \mathbb{R}^{n\times m}$, 我们有

$$Tr(CD) = Tr(DC).$$

解: 假设 $X, A \in \mathbb{R}^{n\times n}$, 我们有

$$\frac{\partial Tr(XAX^T)}{\partial X} = \frac{\partial Tr(XAX_c^T)}{\partial X} + \frac{\partial Tr(X_cAX^T)}{\partial X}$$

$$= (AX_c^T)^T + \frac{\partial Tr(X^TX_cA)}{\partial X}$$

$$= (AX^T)^T + XA = XA^T + XA.$$

结论成立。

迹技巧

在标量微积分中，我们有 $df = f'(x)dx$；类似的，在矩阵微积分中，我们可以定义矩阵函数的微分df。

同时，由公式(3)知，$\frac{\partial Tr(A^TX)}{\partial X} = A$. 所以，如果矩阵微分 df 满足：

$$df = Tr(\mathbf{A}^T d\mathbf{X})$$

那么根据对应关系，直接可以断定：

$$\frac{\partial f}{\partial \mathbf{X}} = \mathbf{A}$$

迹技巧——标准型的简化描述

$$df = Tr(\mathbf{A}^T d\mathbf{X}) \quad \Longrightarrow \quad \frac{\partial f}{\partial \mathbf{X}} = \mathbf{A}$$

计算举例

使用迹技巧计算 $\frac{\partial Tr(\mathbf{BX})}{\partial \mathbf{X}}$。

解：首先，设$f(\mathbf{X}) = Tr(\mathbf{BX})$, 全微分为 $df = d(Tr(\mathbf{BX})) = Tr(d(\mathbf{BX})) = Tr(\mathbf{B}d\mathbf{X})$。
对比标准型的描述：我们需要构造一个 $df = Tr(\mathbf{A}^T d\mathbf{X})$。
显然，只需要设 $\mathbf{A}^T = \mathbf{B}$，即可匹配。所以有 $\frac{\partial \mathrm{tr}(\mathbf{BX})}{\partial \mathbf{X}} = \mathbf{B}^T$。

迹技巧——标准型的简化描述: $df = Tr(\mathbf{A}^T d\mathbf{X}) \quad \Longrightarrow \quad \frac{\partial f}{\partial \mathbf{X}} = \mathbf{A}$

你会求解DAG with no tears的$h(W)$的梯度了吗？

矩阵微分计算 I

***Notears算法中的微分计算**

1. 扩展阅读——矩阵函数在建模中的典型应用——以一篇因果推断论文为例。（矩阵函数用以统计通路数量——请联系《离散数学》图论部分相关结论）

Theorem 1. *A matrix $W \in \mathbb{R}^{d\times d}$ is a DAG if and only if*

$$h(W) = \operatorname{tr}\left(e^{W\circ W}\right) - d = 0, \tag{5}$$

where $\circ$ is the Hadamard product and e^A is the matrix exponential of A. Moreover, $h(W)$ has a simple gradient

$$\nabla h(W) = \left(e^{W\circ W}\right)^T \circ 2W, \tag{6}$$

and satisfies all of the desiderata (a)-(d).

Zheng, Xun, et al. "DAGs with no tears: Continuous optimization for structure learning." *Advances in neural information processing systems* 31 (2018). (Link)

https://arxiv.org/abs/1803.01422

矩阵微分计算 II

***Notears算法中的微分计算**

1. 目标函数与基本定义
目标函数定义为：

$$h(W) = \mathsf{Tr}(e^{W\odot W}) - d$$

其中：$W \in \mathbb{R}^{d\times d}$ 是加权邻接矩阵。$A = W \odot W$ 表示元素级平方（$A_{ij} = W_{ij}^2$）。e^A 是矩阵指数，定义为级数 $e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$。

矩阵微分计算 III

***Notears算法中的微分计算**

2. 梯度的链式法则推导
我们利用全微分来推导 $\nabla h(W)$。
第一步：对矩阵指数的迹求导设 $A = W \odot W$，则 $h(A) = \mathsf{Tr}(e^A) - d$。
根据矩阵分析中的性质，迹的微分与矩阵指数的微分具有以下关系：

$$d(\mathsf{Tr}(e^A)) = \mathsf{Tr}(e^A \cdot dA)$$

因此，关于 A 的梯度为：

$$\nabla_A h(A) = (e^A)^T$$

由于对于邻接矩阵平方构成的 A，其指数矩阵的转置通常直接参与运算，我们可以得到 $\frac{\partial h}{\partial A} = (e^A)^T$。

矩阵微分计算 IV

***Notears算法中的微分计算**

第二步：处理阿达玛积 (Hadamard Product)接下来处理 $A = W \odot W$ 对 W 的导数。根据链式法则，若 h 是关于 A 的函数，且 $A = W \odot W$，则关于 W 的梯度公式为：

$$\nabla_W h = \nabla_A h \odot \frac{d(W \odot W)}{dW}$$

由于 $A_{ij} = W_{ij}^2$，其对 W_{ij} 的导数为 $2W_{ij}$。
第三步：组合最终公式将上述两步结合，得到 $h(W)$ 关于 W 的梯度：

$$\nabla h(W) = (e^{W \odot W})^T \odot 2W$$

矩阵微分计算 V

***Notears算法中的微分计算**

3. 公式总结
根据论文推导，最终梯度表达式为：

$$\nabla h(W) = 2(e^{W \odot W})^T \odot W.$$

你做好设计人工智能算法的准备了吗？