矩阵求导术（上）

长躯鬼侠
数学爱好者

矩阵求导的技术，在统计学、控制论、机器学习等领域有广泛的应用。鉴于我看过的一些资料或言之不详、或繁乱无绪，本文来做个科普，分作两篇，上篇讲标量对矩阵的求导术，下篇讲矩阵对矩阵的求导术。本文使用小写字母x表示标量，粗体小写字母 $\boldsymbol{x}$ 表示（列）向量，大写字母X表示矩阵。

首先来琢磨一下定义，标量f对矩阵X的导数，定义为 $\frac{\partial f}{\partial X} = \left[\frac{\partial f}{\partial X_{ij}}\right]$，即f对X逐元素求导排成与X尺寸相同的矩阵。然而，这个定义在计算中并不好用，实用上的原因是在对较复杂的函数难以逐元素求导；哲理上的原因是逐元素求导破坏了**整体性**。试想，为何要将f看做矩阵X而不是各元素 X_{ij} 的函数呢？答案是用矩阵运算更整洁。所以在求导时不宜拆开矩阵，而是要找一个从整体出发的算法。

为此，我们来回顾，一元微积分中的导数（标量对标量的导数）与微分有联系：$df = f'(x)dx$；多元微积分中的梯度（标量对向量的导数）也与微分有联系：$df = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}dx_i = \frac{\partial f}{\partial \boldsymbol{x}}^T d\boldsymbol{x}$，这里第一个等号是全微分公式，第二个等号表达了梯度与微分的联系：全微分 df 是 $n \times 1$ 梯度向量 $\frac{\partial f}{\partial \boldsymbol{x}}$ 与 $n \times 1$ 微分向量 $d\boldsymbol{x}$ 的内积；受此启发，我们将矩阵导数与微分建立联系：$df = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\partial f}{\partial X_{ij}}dX_{ij} = \text{tr}\left(\frac{\partial f}{\partial X}^T dX\right)$。其中tr代表迹(trace)是方阵对角线元素之和，满足性质：对尺寸相同的矩阵A,B，$\text{tr}(A^TB) = \sum_{i,j}A_{ij}B_{ij}$，即 $\text{tr}(A^TB)$ 是矩阵A,B的**内积**。与梯度相似，这里第一个等号是全微分公式，第二个等号表达了矩阵导数与微分的联系：全微分 df 是 $m \times n$ 导数 $\frac{\partial f}{\partial X}$ 与 $m \times n$ 微分矩阵 dX 的内积。

然后来建立运算法则。回想遇到较复杂的一元函数如 $f = \log(2+\sin x)e^{\sqrt{x}}$，我们是如何求导的呢？通常不是从定义开始求极限，而是先建立了初等函数求导和四则运算、复合等法则，再来运用这些法则。故而，我们来创立常用的矩阵微分的运算法则：

1. 加减法：$d(X \pm Y) = dX \pm dY$；矩阵乘法：$d(XY) = (dX)Y + XdY$；转置：$d(X^T) = (dX)^T$；迹：$d\text{tr}(X) = \text{tr}(dX)$。
2. 逆：$dX^{-1} = -X^{-1}dXX^{-1}$。此式可在 $XX^{-1} = I$ 两侧求微分来证明。
3. 行列式：$d|X| = \text{tr}(X^{\#}dX)$，其中 $X^{\#}$ 表示X的伴随矩阵，在X可逆时又可以写作 $d|X| = |X|\text{tr}(X^{-1}dX)$。此式可用Laplace展开来证明，详见张贤达《矩阵分析与应用》第279页。
4. 逐元素乘法：$d(X \odot Y) = dX \odot Y + X \odot dY$，$\odot$ 表示尺寸相同的矩阵X,Y逐元素相乘。
5. 逐元素函数：$d\sigma(X) = \sigma'(X) \odot dX$，$\sigma(X) = [\sigma(X_{ij})]$ 是逐元素标量函数运算，$\sigma'(X) = [\sigma'(X_{ij})]$ 是逐元素求导数。举个例子，$X = [x_1, x_2], d\sin(X) = [\cos x_1 dx_1, \cos x_2 dx_2] = \cos(X) \odot dX$。

我们试图利用矩阵导数与微分的联系 $df = \text{tr}\left(\frac{\partial f}{\partial X}^T dX\right)$，在求出左侧的微分 df 后，该如何写成右侧的形式并得到导数呢？这需要一些迹技巧(trace trick)：

1. 标量套上迹：

2. 转置：$\text{tr}(A^T) = \text{tr}(A)$。
3. 线性：$\text{tr}(A \pm B) = \text{tr}(A) \pm \text{tr}(B)$。
4. 矩阵乘法交换：$\text{tr}(AB) = \text{tr}(BA)$，其中 A 与 B^T 尺寸相同。两侧都等于 $\sum_{i,j} A_{ij} B_{ji}$。
5. 矩阵乘法/逐元素乘法交换：$\text{tr}(A^T(B \odot C)) = \text{tr}((A \odot B)^T C)$，其中 A, B, C 尺寸相同。两侧都等于 $\sum_{i,j} A_{ij} B_{ij} C_{ij}$。

观察一下可以断言，**若标量函数f是矩阵X经加减乘法、行列式、逆、逐元素函数等运算构成，则使用相应的运算法则对f求微分，再使用迹技巧给df套上迹并将其它项交换至dX左侧，即能得到导数。**

在建立法则的最后，来谈一谈复合：假设已求得 $\frac{\partial f}{\partial Y}$，而Y是X的函数，如何求 $\frac{\partial f}{\partial X}$ 呢？在微积分中有标量求导的链式法则 $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} \frac{\partial y}{\partial x}$，但这里我们不能沿用链式法则，因为矩阵对矩阵的导数 $\frac{\partial Y}{\partial X}$ 截至目前仍是未定义的。于是我们继续追本溯源，链式法则是从何而来？源头仍然是微分。我们直接从微分入手建立复合法则：先写出 $df = \text{tr}\left(\frac{\partial f}{\partial Y}^T dY\right)$，再将dY用dX表示出来代入，并使用迹技巧将其他项交换至dX左侧，即可得到 $\frac{\partial f}{\partial X}$。

接下来演示一些算例。特别提醒要依据已经建立的运算法则来计算，不能随意套用微积分中标量导数的结论，比如认为AX对X的导数为A，这是没有根据、意义不明的。

例1：$f = \boldsymbol{a}^T X \boldsymbol{b}$，求 $\frac{\partial f}{\partial X}$。其中 $\boldsymbol{a}$ 是 $m \times 1$ 列向量，X 是 $m \times n$ 矩阵，$\boldsymbol{b}$ 是 $n \times 1$ 列向量，f 是标量。

解：先使用矩阵乘法法则求微分，这里的 $\boldsymbol{a}, \boldsymbol{b}$ 是常量，$d\boldsymbol{a} = \boldsymbol{0}, d\boldsymbol{b} = \boldsymbol{0}$，得到：$df = \boldsymbol{a}^T dX \boldsymbol{b}$，再套上迹并做矩阵乘法交换：$df = \text{tr}(\boldsymbol{a}^T dX \boldsymbol{b}) = \text{tr}(\boldsymbol{b}\boldsymbol{a}^T dX)$，注意这里我们根据 $\text{tr}(AB) = \text{tr}(BA)$ 交换了 $\boldsymbol{a}^T dX$ 与 $\boldsymbol{b}$。对照导数与微分的联系 $df = \text{tr}\left(\frac{\partial f}{\partial X}^T dX\right)$，得到 $\frac{\partial f}{\partial X} = (\boldsymbol{b}\boldsymbol{a}^T)^T = \boldsymbol{a}\boldsymbol{b}^T$。

注意：这里不能用 $\frac{\partial f}{\partial X} = \boldsymbol{a}^T \frac{\partial X}{\partial X} \boldsymbol{b} = ?$，导数与乘常数矩阵的交换是不合法则的运算（而微分是合法的）。有些资料在计算矩阵导数时，会略过求微分这一步，这是逻辑上解释不通的。

例2【线性回归】：$l = \|X\boldsymbol{w} - \boldsymbol{y}\|^2$，求 $\boldsymbol{w}$ 的最小二乘估计，即求 $\frac{\partial l}{\partial \boldsymbol{w}}$ 的零点。其中 $\boldsymbol{y}$ 是 $m \times 1$ 列向量，X 是 $m \times n$ 矩阵，$\boldsymbol{w}$ 是 $n \times 1$ 列向量，l 是标量。

解：严格来说这是标量对向量的导数，不过可以把向量看做矩阵的特例。先将向量模平方改写成向量与自身的内积：$l = (X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y})$，求微分，使用矩阵乘法、转置等法则：$dl = (Xd\boldsymbol{w})^T(X\boldsymbol{w} - \boldsymbol{y}) + (X\boldsymbol{w} - \boldsymbol{y})^T(Xd\boldsymbol{w}) = 2(X\boldsymbol{w} - \boldsymbol{y})^T X d\boldsymbol{w}$。对照导数与微分的联系 $dl = \frac{\partial l}{\partial \boldsymbol{w}}^T d\boldsymbol{w}$，得到 $\frac{\partial l}{\partial \boldsymbol{w}} = (2(X\boldsymbol{w} - \boldsymbol{y})^T X)^T = 2X^T(X\boldsymbol{w} - \boldsymbol{y})$。$\frac{\partial l}{\partial \boldsymbol{w}}$ 的零点即 $\boldsymbol{w}$ 的最小二乘估计为 $\boldsymbol{w} = (X^T X)^{-1} X^T \boldsymbol{y}$。

例3【多元logistic回归】：$l = -\boldsymbol{y}^T \log \text{softmax}(W\boldsymbol{x})$，求 $\frac{\partial l}{\partial W}$。其中 $\boldsymbol{y}$ 是除一个元素为1外其它元素为0的

$\text{softmax}(\boldsymbol{a}) = \frac{\exp(\boldsymbol{a})}{\mathbf{1}^T\exp(\boldsymbol{a})}$ ，其中 $\exp(\boldsymbol{a})$ 表示逐元素求指数，$\mathbf{1}$ 代表全1向量。

解：首先将softmax函数代入并写成 $l = -\boldsymbol{y}^T\left(\log(\exp(W\boldsymbol{x})) - \mathbf{1}\log(\mathbf{1}^T\exp(W\boldsymbol{x}))\right) = -\boldsymbol{y}^TW\boldsymbol{x} + \log(\mathbf{1}^T\exp(W\boldsymbol{x}))$ ，这里要注意逐元素log满足等式 $\log(\boldsymbol{u}/c) = \log(\boldsymbol{u}) - \mathbf{1}\log(c)$ ，以及 $\boldsymbol{y}$ 满足 $\boldsymbol{y}^T\mathbf{1} = 1$ 。求微分，使用矩阵乘法、逐元素函数等法则：$dl = -\boldsymbol{y}^TdW\boldsymbol{x} + \frac{\mathbf{1}^T\left(\exp(W\boldsymbol{x})\odot(dW\boldsymbol{x})\right)}{\mathbf{1}^T\exp(W\boldsymbol{x})}$ 。再套上迹并做交换，注意可化简 $\mathbf{1}^T\left(\exp(W\boldsymbol{x})\odot(dW\boldsymbol{x})\right) = \exp(W\boldsymbol{x})^TdW\boldsymbol{x}$ ，这是根据等式 $\mathbf{1}^T(\boldsymbol{u}\odot\boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}$ ，故 $dl = \text{tr}\left(-\boldsymbol{y}^TdW\boldsymbol{x} + \frac{\exp(W\boldsymbol{x})^TdW\boldsymbol{x}}{\mathbf{1}^T\exp(W\boldsymbol{x})}\right) = \text{tr}(\boldsymbol{x}(\text{softmax}(W\boldsymbol{x}) - \boldsymbol{y})^TdW)$ 。对照导数与微分的联系，得到 $\frac{\partial l}{\partial W} = (\text{softmax}(W\boldsymbol{x}) - \boldsymbol{y})\boldsymbol{x}^T$ 。

另解：定义 $\boldsymbol{a} = W\boldsymbol{x}$ ，则 $l = -\boldsymbol{y}^T\log\text{softmax}(\boldsymbol{a})$ ，先如上求出 $\frac{\partial l}{\partial \boldsymbol{a}} = \text{softmax}(\boldsymbol{a}) - \boldsymbol{y}$ ，再利用复合法则：$dl = \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}}^Td\boldsymbol{a}\right) = \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}}^TdW\boldsymbol{x}\right) = \text{tr}\left(\boldsymbol{x}\frac{\partial l}{\partial \boldsymbol{a}}^TdW\right)$ ，得到 $\frac{\partial l}{\partial W} = \frac{\partial l}{\partial \boldsymbol{a}}\boldsymbol{x}^T$ 。

例4【方差的最大似然估计】：样本 $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \sim N(\boldsymbol{\mu}, \Sigma)$ ，求方差 Σ 的最大似然估计。写成数学式是：$l = \log|\Sigma| + \frac{1}{n}\sum_{i=1}^n(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\Sigma^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$ ，求 $\frac{\partial l}{\partial \Sigma}$ 的零点。其中 $\boldsymbol{x}_i$ 是 $m\times 1$ 列向量，$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n\boldsymbol{x}_i$ 是样本均值，Σ 是 $m\times m$ 对称正定矩阵，l 是标量。

解：首先求微分，使用矩阵乘法、行列式、逆等运算法则，第一项是 $d\log|\Sigma| = |\Sigma|^{-1}d|\Sigma| = \text{tr}(\Sigma^{-1}d\Sigma)$ ，第二项是 $\frac{1}{n}\sum_{i=1}^n(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^Td\Sigma^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) = -\frac{1}{n}\sum_{i=1}^n(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\Sigma^{-1}d\Sigma\Sigma^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$ 。再给第二项套上迹做交换：

$$\text{tr}\left(\frac{1}{n}\sum_{i=1}^n(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\Sigma^{-1}d\Sigma\Sigma^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\right) = \frac{1}{n}\sum_{i=1}^n\text{tr}\left((\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\Sigma^{-1}d\Sigma\Sigma^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})\right)$$

MathJax maximum macro substitution count exceeded; is there a recursive macro c

，其中先交换迹与求和，然后将 $\Sigma^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$ 交换到左边，最后再交换迹与求和，并定义 $S = \frac{1}{n}\sum_{i=1}^n(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$ 为样本方差矩阵。得到 $dl = \text{tr}\left(\left(\Sigma^{-1} - \Sigma^{-1}S\Sigma^{-1}\right)d\Sigma\right)$ 。对照导数与微分的联系，有 $\frac{\partial l}{\partial \Sigma} = (\Sigma^{-1} - \Sigma^{-1}S\Sigma^{-1})^T$ ，其零点即 Σ 的最大似然估计为 $\Sigma = S$ 。

最后一例留给经典的神经网络。神经网络的求导术是学术史上的重要成果，还有个专门的名字叫做BP算法，我相信如今很多人在初次推导BP算法时也会颇费一番脑筋，事实上使用矩阵求导术来推导并不复杂。为简化起见，我们推导二层神经网络的BP算法。

例5【二层神经网络】：$l = -\boldsymbol{y}^T\log\text{softmax}(W_2\sigma(W_1\boldsymbol{x}))$ ，求 $\frac{\partial l}{\partial W_1}$ 和 $\frac{\partial l}{\partial W_2}$ 。其中 $\boldsymbol{y}$ 是除一个元素为1外其它元素为0的的 $m\times 1$ 列向量，W_2 是 $m\times p$ 矩阵，W_1 是 $p\times n$ 矩阵，$\boldsymbol{x}$ 是 $n\times 1$ 列向量，l 是标量；$\text{softmax}(\boldsymbol{a}) = \frac{\exp(\boldsymbol{a})}{\mathbf{1}^T\exp(\boldsymbol{a})}$ 同例3，$\sigma(\cdot)$ 是逐元素sigmoid函数 $\sigma(a) = \frac{1}{1+\exp(-a)}$ 。

解：定义 $\boldsymbol{a}_1 = W_1\boldsymbol{x}$，$\boldsymbol{h}_1 = \sigma(\boldsymbol{a}_1)$，$\boldsymbol{a}_2 = W_2\boldsymbol{h}_1$，则 $l = -\boldsymbol{y}^T\log\text{softmax}(\boldsymbol{a}_2)$。在例3中已求出 $\frac{\partial l}{\partial \boldsymbol{a}_2} = \text{softmax}(\boldsymbol{a}_2) - \boldsymbol{y}$。使用复合法则，注意此处 $\boldsymbol{h}_1, W_2$ 都是变量：

$dl = \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}_2}^T d\boldsymbol{a}_2\right) = \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}_2}^T dW_2\boldsymbol{h}_1\right) + \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}_2}^T W_2 d\boldsymbol{h}_1\right)$，使用矩阵乘法交换的迹技巧从第一项得到 $\frac{\partial l}{\partial W_2} = \frac{\partial l}{\partial \boldsymbol{a}_2}\boldsymbol{h}_1^T$，从第二项得到 $\frac{\partial l}{\partial \boldsymbol{h}_1} = W_2^T\frac{\partial l}{\partial \boldsymbol{a}_2}$。接下来求 $\frac{\partial l}{\partial \boldsymbol{a}_1}$，继续使用复合法则，并利用矩阵乘法和逐元素乘法交换的迹技巧：

$\text{tr}\left(\frac{\partial l}{\partial \boldsymbol{h}_1}^T d\boldsymbol{h}_1\right) = \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{h}_1}^T (\sigma'(\boldsymbol{a}_1)\odot d\boldsymbol{a}_1)\right) = \text{tr}\left(\left(\frac{\partial l}{\partial \boldsymbol{h}_1}\odot\sigma'(\boldsymbol{a}_1)\right)^T d\boldsymbol{a}_1\right)$，得到 $\frac{\partial l}{\partial \boldsymbol{a}_1} = \frac{\partial l}{\partial \boldsymbol{h}_1}\odot\sigma'(\boldsymbol{a}_1)$。为求 $\frac{\partial l}{\partial W_1}$，再用一次复合法则：

$\text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}_1}^T d\boldsymbol{a}_1\right) = \text{tr}\left(\frac{\partial l}{\partial \boldsymbol{a}_1}^T dW_1\boldsymbol{x}\right) = \text{tr}\left(\boldsymbol{x}\frac{\partial l}{\partial \boldsymbol{a}_1}^T dW_1\right)$，得到 $\frac{\partial l}{\partial W_1} = \frac{\partial l}{\partial \boldsymbol{a}_1}\boldsymbol{x}^T$。

下篇见zhuanlan.zhihu.com/p/24...。

编辑于 04:57

机器学习 矩阵分析 优化

165 条评论　切换为时间排序

写下你的评论...

方不觉　1 年前

nbnb，一直对矩阵求导感觉无从下手，这几条法则比背公式好记多了！

2

resurrectcore　1 年前

网上有个pdf叫做 The Matrix Cookbook.

17

覃含章　1 年前

写的很清楚，很实用！

2

Liwei C

我看过，相信写这篇文章的人也看过。感觉这里的方法比硬背公式简单，而且对于搞机器学习的这里的够用了

10 查看对话

紫杉 1 年前

期待下集。。我都忘了很多这部分内容了

3

ThinkingCat 1 年前

写的真好，醍醐灌顶

2

于双海 1 年前

期待下集 nbnb

赞

王赟 Maigo 1 年前

赞啊！终于找到系统的方法了！

12

独孤阿毛 1 年前

你好~请问一下，这是对矩阵函数求导，还是对函数矩阵求导?

赞

zzzzz 1 年前

行列式微分的那个可以把行列式放进tr里变成伴随矩阵。行列式对其中元素的偏导显然就是其代数余子式，应该不用行列式不为0。这公式叫jacobi's formula

1

maja 1 年前

标量函数对矩阵全微分定义其实不完善，因为只体现了 标量 由n^2变量组成的一个定量关系 而未体现 标量的自变量按矩阵格式排列. 而且用迹定义不好，原因是它的运算量远大于逐元计算

3

粑卜狸 1 年前

第一个定义式的中括号（及其下标ij）应该在左边吧

1

长躯鬼侠 (作者) 回复 zzzzz 1 年前

嗯，你说得对。

赞 查看对话

长躯鬼侠 (作者) 回复 粑卜狸 1 年前

为了避免混淆，我去掉了下标ij。

赞 查看对话

甄景贤 1 年前

这个超有用，功德无量 :)

赞

maja 1 年前

没有回应，可能你没看懂我的意思：这样吧我们从梯度下降流来考虑：

1）全微分的出发点是没有问题的，但用迹是没有解决问题的，问题的本质是如何定义符号“df / dX”，f必然是

dX 为矩阵形式就决定了 df / dX 。如果用矩阵内积运算 则 就确定了 df / dX 的矩阵形式 衡量的就是df 对于dX变化率.

回到基础问题一 f = a^{T} * X * b 现在我们要根据 df = df/dX • dX 来求解. 所以要确定df的形式 矩阵乘法始终是一个记号！ 和行列式通过伴随矩阵建立联系. 通俗的运算就是加 乘 = 定义记号-累乘-S(i, a)=\sum_{i=1}^{n} a_i 则 f=S(i,a)S(k,x_ik * b_k) 然后两边对X诸元素全微分，df=S(<i,k>, dX){S(p, a) * S(q, b)} . 提取公因式，df = a^T * b ? dX • 1

根据等号原理和恒等法则，则（等号有可能只表示一个解，而非所有解）：

df / dX = a^T * b • 1 其中1是元素都为1的矩阵.

这个是我们可以用矩阵描述梯度下降流的朴素原理，现在考虑第二个问题：

这就是为什呢题主用迹会出错，因为就不对.

👍 1

maja 1 年前

2）自变量长像是矩阵，如果标量可以看成1*1矩阵 矩阵可以看成向量，每个向量元素又是一个向量；向量没有行向量和列向量之分------我们需要定义一个兼容运算格式，这比背公式更有意义：

从梯度开始，我们定义df / dv 行矩阵 还是 列呢?

先上结论，兼容定式1：若df / dv为行向量，则 df / dv^T必为列向量！ 其中v为列向量. 反之亦然.

这个法则可以保证 df / dX 是一个和上述定义相符的矩阵. 若 X = {X(i,j)} = [X_1, X_2, ...]

回到基础问题2: f = x^T *A* x. 求标量二次型f 对矩阵 X在内积定义下的变化率. 同样 df = df / dx • dx, 我们吧 df 展开成某dX 与某个函数的内积 根据f任意性的恒等原理来求解：如前定义，f = S(i, x)S(k, A_ik * x), df/dx_i= S(k, A_ik * x_k) + S(k, x_k * A_ki) = (Ax)_i + (A^Tx)_i

hence, df / dx = x^T(A+A^T) • dx

仔细观察，df / dx 确实是行向量.

最后一个问题 利用已知运算扩展运算，扩展标量函数到矢量函数，这个涉及微分几何的微分映照以后聊.

👍 赞

长躯鬼侠 (作者) 回复 maja 1 年前

我确实没看懂你想表达什么，你说的没问题，但和我有什么矛盾嘛?

👍 赞 查看对话

长躯鬼侠 (作者) 回复 maja 1 年前

tr(A^TB)是矩阵A,B内积的定义，你是对此有异议?

👍 赞 查看对话

maja 回复 长躯鬼侠 (作者) 1 年前

没必要的 tr 的计算量远大于 内积， 他就是个符号，只是方便书写，既不方便推导也不没有太大意义

👍 赞 查看对话