**矩阵微分**

**预先规定**

A’和都代表矩阵转置

ξ代表标量，a代表列向量，A代表矩阵

Φ()代表标量方程输出为标量，f()代表向量方程输出为列向量，F()代表矩阵方程输出为矩阵

矩阵A的shape=(m,n)；矩阵B的shape=(p,q)

列向量化

行向量化

**一、矩阵布局**

Jacobian矩阵采用两种布局，原始布局和分子布局Numerator layout；梯度矩阵采用分母布局Denominator layout。

要注意到并非所有数学教科书和论文在矩阵微分表示方面始终如一。也就是说，有时在同一本书或纸上的不同背景中使用不同的约定。例如，有些选择表示梯度矩阵时使用分母布局，而表示vector-by-vector微分矩阵时使用分子布局

**不同的布局对应的矩阵求导公式不同，求导结果形状也不同。**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Scalar φ | | Vector y(size m×1) | | Matrix Y(size m×n) | |
|  |  | Notation | Shape | Notation | Shape | Notation | Shape |
| Scalar ξ | 原始布局 |  | scalar |  | m×1 |  | mn×1 |
| 分子布局 | m×1 | m×n |
| 分母布局 | 1×m |  |
| Vector x  (size p×1) | 原始布局 |  | 1×p |  | m×p |  | mn×p |
| 分子布局 | 1×p | m×p |  |
| 分母布局 | p×1 | p×m |  |
| Matrix X  (size p×q) | 原始布局 |  | 1×pq |  | m×pq |  | mn×pq |
| 分子布局 | q×p |  |  |
| 分母布局 | p×q |  |  |

注：表格来自https://en.wikipedia.org/wiki/Matrix\_calculus

原始布局：分子(求导公式)决定行数，分母(求导变量)决定列数。

分子布局：不涉及Matrix的情况下，分子(求导公式)决定行数，分母(求导变量)决定列数

分母布局：不涉及Matrix的情况下，分子(求导公式)决定列数，分母(求导变量)决定行数

**二、Jacobian布局细节**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Scalar variable | Vector variable | Matrix variable |
| Scalar function | φ(ξ) | φ(x) | φ(X) |
| Vector function | f(ξ) | f(x) | f(X) |
| Matrix function | F(ξ) | F(x) | F(X) |

1、标量和向量

dφ(ξ)/dξ即 标量/标量*——>* 标量

dφ(x)/ dx即 标量/向量

标量/列向量*——>*行向量

标量/行向量*——>*行向量

df(ξ)/ dξ即 向量/标量

列向量/标量*——>*列向量

行向量/标量*——>*列向量

df(x)/ dx即 向量/向量，

列向量/列向量*——>*

列向量/行向量*——>*

行向量/列向量*——>*

行向量/行向量*——>*

注：——>指输出类型

2、标量和矩阵

矩阵/标量*——>*

原始布局

分子布局

标量/矩阵*——>*

原始布局

分子布局

3、向量和矩阵

矩阵/矩阵*——>*

**三、原始布局下求导**

1、公式知识

[1]矩阵基本运算

[2]矩阵的内积与范数

[3]矩阵的迹

tr (A+B)=tr A + tr B

tr AB=tr BA

[4]矩阵的Kronecker积/张量积

[5]向量化与矩阵化

[6] 一阶实矩阵微分与Jacobian矩阵辨识

2、核心知识整理

[1] Jacobian矩阵定义(扩展到输入为矩阵，输出为矩阵)：

[2] 矩阵微分与Jacobian矩阵的关系:

[3] Jacobian矩阵与梯度矩阵的关系:

3、求导公式

[1]矩阵求导公式

|  |  |
| --- | --- |
| F(X) | Jacobian矩阵DF(X) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

公式来自于《矩阵分析与应用 第二版》P163

[2]一般求导公式

不涉及矩阵的求导公式参考维基百科en.wikipedia.org/wiki/Matrix\_calculus

4、一般Jacobian矩阵方法（推求导公式的方法）

[1] 按Jacobian矩阵定义展开，逐个元素求导

[2] 公式求Jacobian矩阵DxΦ(X)的步骤

step1：将 dΦ(X)转化为迹的形式

step2：由dΦ(X)得到DxΦ(X)

[3] 公式求Jacobian矩阵DxF(X)的步骤

step1：求dF(X)

step2：计算d vec F(X)

step3：由d vec F(X)得DxF(X)和∇xF(X)

5、使导数的形状和自变量的形状一致

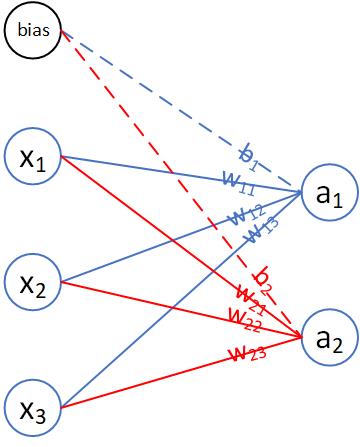
[1]

注意到在原始布局中，自变量(分子)数目和Jacobian矩阵列数一致，为了使导数的形状和自变量的形状一致，我们可以将Jacobian矩阵压缩为一行，对应python语句gradient\_vec = np.sum(Jacobian, axis=0)

进一步，还可以将gradient行向量矩阵化，对应python语句gradient = gradient\_vec.reshape(p,-1,order=’F’)

[2]举例

以一个二维向量输出为例



实际中我们使用gradient([y1,y2], [x1,x2])函数，gradients(y, x)求导y=f(x), 返回一个list，列表长度和x数量一致，列表中每个元素是对于y进行sum(dy/dx)。由于对y进行sum操作，所以能保证维度和梯度分子一致。

对于节点求导：

对于权重求导

6、原始布局求导的缺点

原始布局求导公式复杂，求导结果不够直观。难以人工验证中间计算细节

举例：

可以看到结果很不直观

从结果中得到A也很麻烦

**四、满足shape convention(混合布局)的element-wise矩阵求导**

1、Shape Convention：使导数的形状和自变量的形状一致（以便于梯度下降运算）

2、shape convention两个规则：①改变变元形状②改变导数形状

[1]改变变元形状：

当变元是列变量时，我们是对其转置求偏导；当变元非列变量时，我们对其本身求偏导。

[2]改变导数形状：

Scalar-by-vector时使用分子布局

Vector-by-vector时采用分子布局

Scalar-by-matrix时采用分母布局

3、element-wise矩阵求导

element-wise矩阵求导：对矩阵元素求导，再拼接成矩阵（为了确定shape convention下的求导规则，我们在element-wise层面使用求导规则和链式法则）

element-wise矩阵求导规则见

<http://10.3.200.202/cache/10/03/web.stanford.edu/8e61bd0b459595c3aec3c97fbdee9c74/gradient-notes.pdf>

在element-wise层面使用求导规则和链式法则使求导结果更清晰，求导难度也低于原始布局下的矩阵求导

**五、参考书籍和博客**

<http://10.3.200.202/cache/10/03/web.stanford.edu/8e61bd0b459595c3aec3c97fbdee9c74/gradient-notes.pdf>

<https://zhuanlan.zhihu.com/p/32368246>

《矩阵分析与应用 第二版》张贤达

《Matrix Differential Calculus with Applications in Statistics and Econometrics 3rd Edition》

<https://en.wikipedia.org/wiki/Matrix_calculus>

**六、矩阵微分代码实现**

1、通过element-wise矩阵求导得到满足shape convention的求导结果数学公式

2、对输入的格式进行统一处理，以免外部输入shape不一致导致的问题

3、比较代码中输入的shape和数学公式中各阶段结果的shape，相应调整求导公式，使代码最后输出shape和数学公式一致