

刘建平Pinard

十五年博客，对数学统计学，数据挖掘，机器学习，大数据平台，大数据平台应用开发，大数据可视化感兴趣。

博客园

首页

新随笔

联系

订阅

管理

机器学习中的矩阵向量求导(四) 矩阵向量求导链式法则

在机器学习中的矩阵向量求导(三) 矩阵向量求导之微分法中，我们讨论了使用微分法来求解矩阵向量求导的方法。但是很多时候，求导的自变量和因变量直接有复杂的多层链式求导的关系，此时微分法使用起来也有些麻烦。需要一些简洁的方法。

本文我们讨论矩阵向量求导链式法则，使用该法则很多时候可以帮我们快速求出导数结果。

本文的标量对向量的求导，标量对矩阵的求导使用分母布局， 向量对向量的求导使用分子布局。如果遇到其他资料求导结果不同，请先确认布局是否一样。

1. 向量对向量求导的链式法则

首先我们来看看向量对向量求导的链式法则。假设多个向量存在依赖关系，比如三个向量 $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{z}$ 存在依赖关系，则我们有下面的链式求导法则：

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

该法则也可以推广到更多的向量依赖关系。但是要注意的是要求所有有依赖关系的变量都是向量，如果有一个 \mathbf{Y} 是矩阵，，比如是 $\mathbf{x} \rightarrow \mathbf{Y} \rightarrow \mathbf{z}$ ，则上式并不成立。

从矩阵维度相容的角度也很容易理解上面的链式法则，假设 $\mathbf{x}, \mathbf{y}, \mathbf{z}$ 分别是 m, n, p 维向量，则求导结果 $\frac{\partial \mathbf{z}}{\partial \mathbf{x}}$ 是一个 $p \times m$ 的雅克比矩阵，而右边 $\frac{\partial \mathbf{z}}{\partial \mathbf{y}}$ 是一个 $p \times n$ 的雅克比矩阵， $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ 是一个 $n \times m$ 的矩阵，两个雅克比矩阵的乘积维度刚好是 $p \times m$ ，和左边相容。

2. 标量对多个向量的链式求导法则

在我们的机器学习算法中，最终要优化的一般是一个标量损失函数，因此最后求导的目标是标量，无法使用上一节的链式求导法则，比如2向量，最后到1标量的依赖关系： $\mathbf{x} \rightarrow \mathbf{y} \rightarrow z$ ，此时很容易发现维度不相容。

假设 \mathbf{x}, \mathbf{y} 分别是 m, n 维向量，那么 $\frac{\partial z}{\partial \mathbf{x}}$ 的求导结果是一个 $m \times 1$ 的向量，而 $\frac{\partial z}{\partial \mathbf{y}}$ 是一个 $n \times 1$ 的向量， $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ 是一个 $n \times m$ 的雅克比矩阵,右边的向量和矩阵是没法直接乘的。

但是假如我们把标量求导的部分都做一个转置，那么维度就可以相容了，也就是：

$$(\frac{\partial z}{\partial \mathbf{x}})^T = (\frac{\partial z}{\partial \mathbf{y}})^T \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

但是毕竟我们要求导的是 $(\frac{\partial z}{\partial \mathbf{x}})$,而不是它的转置，因此两边转置我们可以得到标量对多个向量求导的链式法则：

$$\frac{\partial z}{\partial \mathbf{x}} = (\frac{\partial \mathbf{y}}{\partial \mathbf{x}})^T \frac{\partial z}{\partial \mathbf{y}}$$

如果是标量对更多的向量求导,比如 $\mathbf{y}_1 \rightarrow \mathbf{y}_2 \rightarrow \dots \rightarrow \mathbf{y}_n \rightarrow z$ ，则其链式求导表达式可以表示为：

$$\frac{\partial z}{\partial \mathbf{y}_1} = (\frac{\partial \mathbf{y}_n}{\partial \mathbf{y}_{n-1}} \frac{\partial \mathbf{y}_{n-1}}{\partial \mathbf{y}_{n-2}} \dots \frac{\partial \mathbf{y}_2}{\partial \mathbf{y}_1})^T \frac{\partial z}{\partial \mathbf{y}_n}$$

这里我们给一个最常见的最小二乘法求导的例子。最小二乘法优化的目标是最小化如下损失函数：

$$l = (X\theta - y)^T (X\theta - y)$$

我们优化的损失函数 l 是一个标量，而模型参数 θ 是一个向量，期望 l 对 θ 求导，并求出导数等于0时候的极值点。我们假设向量 $z = X\theta - y$ ，则 $l = z^T z$ ， $\theta \rightarrow z \rightarrow l$ 存在链式求导的关系，因此：

$$\frac{\partial l}{\partial \theta} = (\frac{\partial z}{\partial \theta})^T \frac{\partial l}{\partial z} = X^T (2z) = 2X^T (X\theta - y)$$

其中最后一步转换使用了如下求导公式：

$$\begin{aligned} \frac{\partial X\theta - y}{\partial \theta} &= X \\ \frac{\partial z^T z}{\partial z} &= 2z \end{aligned}$$

这两个式子我们在前几篇里已有求解过，现在可以直接拿来使用了，非常方便。

当然上面的问题使用微分法求导数也是非常简单的，这里只是给出链式求导法的思路。

3. 标量对多个矩阵的链式求导法则

下面我们再来看看标量对多个矩阵的链式求导法则，假设有这样的依赖关系： $\mathbf{X} \rightarrow \mathbf{Y} \rightarrow z$,那么我们有：

公告

★珠江追梦，饮岭南茶，恋鄂比酒
你的支持是我写作的动力：



昵称： 刘建平Pinard
园龄： 6年1个月
粉丝： 9622
关注： 15
+加关注

积分与排名

积分 - 487966
排名 - 1185

随笔分类 (135)

0040. 数学统计学(9)
0081. 机器学习(71)
0082. 深度学习(11)
0083. 自然语言处理(23)
0084. 强化学习(19)
0121. 大数据挖掘(1)
0122. 大数据平台(1)

随笔档案 (135)

2019年7月(1)
2019年6月(1)
2019年5月(2)
2019年4月(3)
2019年3月(2)
2019年2月(2)
2019年1月(2)
2018年12月(1)
2018年11月(1)
2018年10月(3)
2018年9月(3)
2018年8月(4)
2018年7月(3)
2018年6月(3)
2018年5月(3)
更多

常去的机器学习网站

强化学习入门书
52 NLP
Analytics Vidhya
深度学习进阶书
深度学习入门书
机器学习路线图
机器学习库

阅读排行榜

$$\frac{\partial z}{\partial x_{ij}} = \sum_{k,l} \frac{\partial z}{\partial Y_{kl}} \frac{\partial Y_{kl}}{\partial X_{ij}} = tr((\frac{\partial z}{\partial Y})^T \frac{\partial Y}{\partial X_{ij}})$$

这里大家会发现我们没有给出基于矩阵整体的链式求导法则，主要原因是矩阵对矩阵的求导是比较复杂的定义，我们目前也未涉及。因此只能给出对矩阵中一个标量的链式求导方法。这个方法并不实用，因为我们并不想每次都基于定义来求导最后再去排列求导结果。

虽然我们没有全局的标量对矩阵的链式求导法则，但是对于一些线性关系的链式求导，我们还是可以得到一些有用的结论的。

我们来看这个常见问题： A, X, B, Y 都是矩阵， z 是标量，其中 $z = f(Y), Y = AX + B$,我们要求出 $\frac{\partial z}{\partial X}$,这个问题在机器学习中是很常见的。此时，我们并不能直接整体使用矩阵的链式求导法则，因为矩阵对矩阵的求导结果不好处理。

这里我们回归初心，使用定义法试一试,先使用上面的标量链式求导公式：

$$\frac{\partial z}{\partial x_{ij}} = \sum_{k,l} \frac{\partial z}{\partial Y_{kl}} \frac{\partial Y_{kl}}{\partial X_{ij}}$$

我们再来看看后半部分的导数：

$$\frac{\partial Y_{kl}}{\partial X_{ij}} = \frac{\partial \sum_s (A_{ks}X_{sl})}{\partial X_{ij}} = \frac{\partial A_{ki}X_{il}}{\partial X_{ij}} = A_{ki}\delta_{lj}$$

其中 δ_{lj} 在 $l = j$ 时为1，否则为0。

那么最终的标签链式求导公式转化为：

$$\frac{\partial z}{\partial x_{ij}} = \sum_{k,l} \frac{\partial z}{\partial Y_{kl}} A_{ki}\delta_{lj} = \sum_k \frac{\partial z}{\partial Y_{kj}} A_{ki}$$

即矩阵 A^T 的第*i*行和 $\frac{\partial z}{\partial Y}$ 的第*j*列的内积。排列成矩阵即为：

$$\frac{\partial z}{\partial X} = A^T \frac{\partial z}{\partial Y}$$

总结下就是：

$$z = f(Y), Y = AX + B \rightarrow \frac{\partial z}{\partial X} = A^T \frac{\partial z}{\partial Y}$$

这结论在 \mathbf{x} 是一个向量的时候也成立，即：

$$z = f(\mathbf{y}), \mathbf{y} = A\mathbf{x} + \mathbf{b} \rightarrow \frac{\partial z}{\partial \mathbf{x}} = A^T \frac{\partial z}{\partial \mathbf{y}}$$

如果要求导的自变量在左边，线性变换在右边，也有类似稍有不同结论如下，证明方法是类似的，这里直接给出结论：

$$z = f(Y), Y = XA + B \rightarrow \frac{\partial z}{\partial X} = \frac{\partial z}{\partial Y} A^T$$

$$z = f(\mathbf{y}), \mathbf{y} = X\mathbf{a} + \mathbf{b} \rightarrow \frac{\partial z}{\partial \mathbf{X}} = \frac{\partial z}{\partial \mathbf{y}} \mathbf{a}^T$$

使用好上述四个结论，对于机器学习尤其是深度学习里的求导问题可以非常快的解决,大家可以试一试。

4. 矩阵向量求导小结

矩阵向量求导在前面我们讨论三种方法，定义法，微分法和链式求导法。在同等情况下，优先考虑链式求导法，尤其是第三节的四个结论。其次选择微分法、在没有好的求导方法的时候使用定义法是最后的保底方案。

基本上大家看了系列里这四篇后对矩阵向量求导就已经很熟悉了，对于机器学习出现的矩阵向量求导问题已足够。这里还没有讲到的是矩阵对矩阵的求导，还有矩阵对向量，向量对矩阵求导这三种形式，这个我们在系列的下一篇，也是最后一篇简单讨论一下，如果大家只是关注机器学习的优化问题，不涉及其他应用数学问题的，可以不关注。

(欢迎转载，转载请注明出处。欢迎沟通交流：liujianping-ok@163.com)

分类: 0040. 数学统计学

标签: 矩阵求导 向量求导

好文置顶

关注我

收藏该文



刘建平Pinard
粉丝 - 9622 关注 - 15

±加关注

« 上一篇: 机器学习中的矩阵向量求导(三) 矩阵向量求导之微分法
» 下一篇: 机器学习中的矩阵向量求导(五) 矩阵对矩阵的求导

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页

-
1. ;

2. ;

3. |

4. ;

43. ;

5. ;

评论排行榜

1. 梯度提升树(GBDT)原理小结
2. 集成学习之Adaboost算法原
3. 决策树算法原理(下)(341)
4. 强化学习(十六) 深度确定性
- 16)
5. 谱聚类 (spectral clusterin

推荐排行榜

1. 梯度下降 (Gradient Desce
2. 奇异值分解(SVD)原理与在
3. 谱聚类 (spectral clusterin
4. 集成学习之Adaboost算法原
5. 梯度提升树(GBDT)原理小结

【推荐】阿里云金秋云创季，云服务器2核2G低至49.68元/年

【推荐】双十一同价！腾讯云云服务器抢先购，低至4.2元/月

编辑推荐：

- 聊一聊如何截获 C# 程序产生的日志
- 一步一图带你深入理解 Linux 物理内存管理
- 快速构建页面结构的 3D Visualization
- 技术管理之如何协调加班问题
- 新零售 SaaS 架构：多租户系统架构设计

阅读排行：

- 浏览器打印方案
- 推荐一款 .NET 编写的 嵌入式平台的开源仿真器--Renode
- 自动注册实体类到EntityFramework Core上下文，并适配ABP及ABP VNext
- 在C#中使用Halcon开发视觉检测程序
- 【动手学深度学习】学习笔记