前反向传播与导数计算

宋奕龙　袁坤

2024 年 10 月 16 日

1　数学分析中的导数

1.1　多元单值函数的梯度

定义 1.1. 设 $f(\boldsymbol{x}):\mathbb{R}^d\to\mathbb{R}$，定义梯度

$$\nabla f(\boldsymbol{x})=\left(\frac{\partial f}{\partial x_1},\dots,\frac{\partial f}{\partial x_d}\right)\in\mathbb{R}^{1\times d} \tag{1}$$

此时梯度为一个行向量，是为了方便如下式子成立：

$$\mathrm{d}f=\left(\frac{\partial f}{\partial x_1},\dots,\frac{\partial f}{\partial x_d}\right)\begin{pmatrix}\mathrm{d}x_1\\ \vdots\\ \mathrm{d}x_d\end{pmatrix}=\nabla f(\boldsymbol{x})\cdot\mathrm{d}\boldsymbol{x} \tag{2}$$

1.2　多元多值函数的雅各比矩阵

定义 1.2. 设 $\boldsymbol{f}(\boldsymbol{x}):\mathbb{R}^d\to\mathbb{R}^p$，定义它的雅各比矩阵为：

$$\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}}=\begin{pmatrix}\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{x_d}\\ \vdots & & \vdots\\ \frac{\partial f_p}{\partial x_1} & \cdots & \frac{\partial f_p}{\partial x_d}\end{pmatrix}_{p\times d} \tag{3}$$

此时它的列数为自变量的个数，行数为因变量的个数。

1.3 链式法则的矩阵实现

将函数的梯度与雅各比矩阵表示成上面的形式的一个好处是方便用矩阵形式实现链式法则，下面是一个例子：设 $\boldsymbol{y}=\boldsymbol{g}(\boldsymbol{x}):\mathbb{R}^d\to\mathbb{R}^p$，　$f=f(\boldsymbol{y}):\mathbb{R}^p\to\mathbb{R}$，则

$$\frac{\partial f}{\partial \boldsymbol{x}}_{1\times d}=\frac{\partial f}{\partial \boldsymbol{y}}_{1\times p}\cdot\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}_{p\times d} \tag{4}$$

或写作：

$$\mathrm{d}f=\frac{\partial f}{\partial \boldsymbol{y}}_{1\times p}\cdot\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}_{p\times d}\mathrm{d}\boldsymbol{x} \tag{5}$$

可以使用简单例子证明这一结论。

2 机器学习与优化中的导数

2.1 多元单值函数的梯度

由于机器学习中经常使用梯度下降，我们往往更倾向于使梯度与自变量保持相同的形状，因此在机器学习中梯度往往用列向量表示，即使这会为链式法则的使用带来一些麻烦。即：

定义 2.1. 设 $f(\boldsymbol{x}):\mathbb{R}^d\to\mathbb{R}$，定义梯度

$$\nabla f(\boldsymbol{x})=\left(\frac{\partial f}{\partial x_1},\ldots,\frac{\partial f}{\partial x_d}\right)^{\top}\in\mathbb{R}^{d\times 1} \tag{6}$$

因此与之对应的微分形式我们也需做一调整，写成如下形式：

$$\mathrm{d}f=(\mathrm{d}x_1,\ldots,\mathrm{d}x_d)\begin{pmatrix}\frac{\partial f}{\partial x_1}\\ \vdots\\ \frac{\partial f}{\partial x_d}\end{pmatrix}=\mathrm{d}\boldsymbol{x}^{\top}\cdot\nabla f(\boldsymbol{x}) \tag{7}$$

2.2 多元多值函数的导数

在机器学习中，多元多值函数的导数和数学分析中雅各比矩阵的定义是相同的，以下是一个简单的计算例子：

例 2.2. 设 $\boldsymbol{y} = W\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^d, \quad \boldsymbol{y} \in \mathbb{R}^p, \quad W \in \mathbb{R}^{p\times d}$，求其雅各比矩阵。
将等式展开：

$$y_1 = w_{11}x_1 + w_{12}x_2 + \cdots + w_{1d}x_d \tag{8}$$
$$\cdots \tag{9}$$
$$y_p = w_{p1}x_1 + w_{p2}x_2 + \cdots + w_{pd}x_d \tag{10}$$

观察后不难发现

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = W \tag{11}$$

2.3 以矩阵为变量的单值函数的导数

将梯度写为列向量的另一好处是可以和矩阵求导兼容，矩阵导数的定义如下：

定义 2.3. 设 $f(X) : \mathbb{R}^{m\times n} \to \mathbb{R}$，则定义矩阵导数如下：

$$\nabla f(X) = \begin{pmatrix} \frac{\partial f}{\partial x_{11}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \vdots & & \vdots \\ \frac{\partial f}{\partial x_{m1}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{pmatrix} \tag{12}$$

实际上这里就是对矩阵的每个元素求导数后拼成原来矩阵的形状，实际上相比于向量求导并没有引入新事物，矩阵的结构在这里并没有发挥作用。

2.4 向量与矩阵函数求导的极限定义法

定义 2.4 (矩阵的 Frobenius 范数与内积)**.** 1. 设矩阵 $A = (a_{ij}) \in \mathbb{R}^{m\times n}$, 则其 Frobenius 范数定义为：

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_i j} \tag{13}$$

2. 设矩阵 $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, 则其 Frobenius 内积定义为：

$$\langle A, B\rangle_F = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij} = \mathrm{tr}(A^\top B) = \mathrm{tr}(AB^\top) = \mathrm{tr}(B^\top A) = \mathrm{tr}(BA^\top) \tag{14}$$

矩阵的 Frobenius 范数与内积是向量的二范数与内积的自然推广，下面我们给出向量函数导数的极限定义，矩阵导数的极限定义可以类比得到。

定义 2.5 (Gâteaux 导数)**.** 对于向量函数 $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$，若存在向量 $\boldsymbol{g} \in \mathbb{R}^n$ 使得对任意方向向量 $\boldsymbol{v} \in \mathbb{R}^n$, 均有：

$$\lim_{t \to 0} \frac{f(\boldsymbol{x} + t\boldsymbol{v}) - f(\boldsymbol{x})}{t} = \langle \boldsymbol{g}, \boldsymbol{v} \rangle, \tag{15}$$

则称向量 $\boldsymbol{g}$ 为 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}$ 处的 Gâteaux 导数。

Gâteaux 导数避开了求分量的偏导数，而直接对向量或矩阵整体求导数，结合泰勒公式、链式法则可以更好地应对更多求导问题。在实际求解过程中需要灵活运用。

2.5 矩阵导数的处理

引入矩阵变量的一个问题就是它不能与雅各比矩阵和链式法则兼容了，解决这一问题的方法有两种：其一，引入张量以及矩阵的元素点积；其二，将矩阵变量平铺为列向量。我们在这里介绍第二种方法。设 $W = (\vec{\boldsymbol{w}}_1, \ldots, \vec{\boldsymbol{w}}_n)_{m \times n}$，即将其拆成 n 个列向量，随后将其纵向排列，记平铺后的列向量为

$$\mathrm{vec}(W) = \begin{pmatrix} \vec{\boldsymbol{w}}_1 \\ \vdots \\ \vec{\boldsymbol{w}}_n \end{pmatrix}_{(m \times n) \times 1} \tag{16}$$

它是一个 $(m \times n) \times 1$ 维的狭长列向量。对应地我们记平铺后的向量梯度为 $\dfrac{\partial f}{\partial \mathrm{vec}(W)}$，下一节中我们将通过一个具体例子演示其用法。

2.6 一个常用的引理

引理 2.6. 设 $f = f(y), \quad y = Wx, \quad x \in \mathbb{R}^d, \quad W \in \mathbb{R}^{p \times d}, \quad y \in \mathbb{R}^p$，则

$$\frac{\partial f}{\partial W} = \frac{\partial f}{\partial y} x^\top \tag{17}$$

证明如下：

想要使用链式法则，我们需要注意两点：

- 其一，必须将矩阵变量平铺为向量，才可使用链式法则。
- 其二，梯度作为行向量才可参与链式法则。（也就是我们将其放在链式法则中时需要进行转置，这些梯度的分子部分往往是 f, L 等。）

因此链式法则表示为：

$$\left(\frac{\partial f}{\partial \mathrm{vec}(W)} \right)^\top = \left(\frac{\partial f}{\partial y} \right)^\top \frac{\partial y}{\partial \mathrm{vec}(W)} \tag{18}$$

我们只需观察后一个因式即可，事实上观察

$$y_1 = w_{11}x_1 + w_{12}x_2 + \cdots + w_{1d}x_d \tag{19}$$

$$\cdots \tag{20}$$

$$y_p = w_{p1}x_1 + w_{p2}x_2 + \cdots + w_{pd}x_d \tag{21}$$

可以得到：

$$\frac{\partial y}{\partial \mathrm{vec}(W)} = (x_1 I_p, \cdots, x_d I_p)_{p\times(p\times d)} \tag{22}$$

注意这里我们是按照 $w_{11}, w_{21}, \ldots, w_{p1}, w_{12}, \ldots, w_{p2}, \ldots, w_{1d}, \ldots, w_{pd}$ 的顺序平铺展开的，即按列展开，这和上面关于 vec 的定义是一致的。如果按行展开，虽然不会发生错误，但形式会有明显不同，在计算时注意上下文行文统一即可。代入计算可以得到：

$$\frac{\partial f}{\partial \mathrm{vec}(W)} = (x_1 I_p, \cdots, x_d I_p)_{(p\times d)\times p} \cdot \frac{\partial f}{\partial y} \tag{23}$$

$$= \begin{pmatrix} x_1 \frac{\partial f}{\partial y} \\ \vdots \\ x_d \frac{\partial f}{\partial y} \end{pmatrix}_{(p\times d)\times 1} \tag{24}$$

再将向量重排成矩阵得：

$$\frac{\partial f}{\partial W} = \left(x_1 \frac{\partial f}{\partial y}, \ldots, x_d \frac{\partial f}{\partial y} \right)_{p\times d} = \frac{\partial f}{\partial y} x^\top \tag{25}$$

■

可以看出这个证明非常地不直观。事实上，我们今后大多会直接将其作为一个引理来使用。在实际计算时，我们往往会先使用这一引理，将矩阵梯度转化为向量梯度与一个向量的转置乘积的形式，随后再用链式法则计算向量梯度，计算过程中尽量避免对矩阵变量使用链式法则。

3 一个反向传播的典例

3.1 问题设定

考虑如下的神经网络模型：

$$\begin{array}{lll} h_1 = W_1 x & x \in \mathbb{R}^d & W_1 \in \mathbb{R}^{p\times d} \quad h_1 \in \mathbb{R}^p \\ z_1 = \phi(h_1) & z_1 \in \mathbb{R}^p & \\ h_2 = W_2 z_1 & W_2 \in \mathbb{R}^{q\times p} & h_2 \in \mathbb{R}^q \\ z_2 = \phi(h_2) & z_2 \in \mathbb{R}^q & \\ \hat{y} = W_3 z_2 & W_3 \in \mathbb{R}^{s\times q} & \hat{y} \in \mathbb{R}^s \\ f = L(y, \hat{y}) & f \in \mathbb{R} & \end{array}$$

图中黑色为已知量，绿色为前向传播时计算的量，红色为欲求的导数值，蓝色和紫

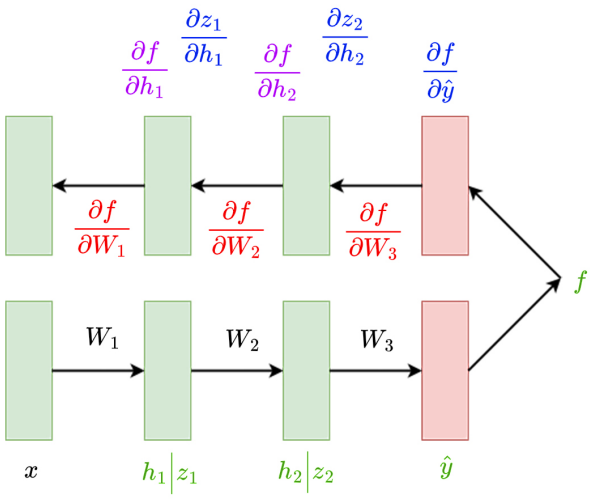

Figure 1: 一个三层神经网络示意图

色均为反向传播计算导数时的中间量，其中蓝色为可以直接计算出导函数，随后代入自变量的值求得的单点导数，紫色为通过链式法则计算的单点导数。我们在下文中将会解释。

3.2 注意事项

首先我们应当注意，对于大型的神经网络，我们求得的导数并不是一个完整的“导函数”，求出完整的导函数几乎不可能。事实上，每一次“前向传播”，即一次函数求值时，W_1, W_2, W_3 都要事先取定一个已知的值，而我们求的导数便是在 W_1, W_2, W_3 给定时的“单点导数值”。且当我们对某一变量求偏导时，其余变量我们当作常数处理。我们依次计算关于 W_3, W_2, W_1 的导数，根据经验，导数的计算大致遵循以下两个步骤：

1. 使用引理将其化为一个向量梯度和一个系数向量乘积的形式。
2. 使用链式法则计算向量梯度，中间过程中避免出现矩阵变量。

3.3 求导过程

由引理 1 可知，关于 W_3 的导数为：

$$\frac{\partial f}{\partial W_3} = \frac{\partial f}{\partial \hat{y}} z_2^\top \tag{26}$$

其中，z_2 在前向传播时已经计算过，蕴含着 W_1, W_2 的信息，但我们将其作为常数处理；$\frac{\partial f}{\partial \hat{y}}$ 是 $\frac{\partial f}{\partial \hat{y}}\Big|_{\hat{y}}$ 的简写，导函数可以直接计算出来，随后代入前向传播时 $\hat{y}$ 的值即可。

计算关于 W_2 的导数时，首先令：

$$\frac{\partial f}{\partial W_2} = \frac{\partial f}{\partial h_2} z_1^\top \tag{27}$$

z_1 已知，而 $\dfrac{\partial f}{\partial h_2}$ 需要用链式法则计算。注意到它在这里是一个列向量，因此我们需要将其转为行向量才可使用链式法则。

$$\left(\frac{\partial f}{\partial h_2}\right)^\top = \left(\frac{\partial f}{\partial \hat{y}}\right)^\top \frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial h_2} \tag{28}$$

其中 $\left(\dfrac{\partial f}{\partial \hat{y}}\right)^\top$ 在求 W_3 的导数时已经求过。$\dfrac{\partial \hat{y}}{\partial z_2}$ 是线性映射，其取值就是 W_3，而在求 W_2 的导数时，W_3 视为常数。$\dfrac{\partial z_2}{\partial h_2}$ 是一个激活函数层，通常激活函数都是元素级别的、标量到标量的映射，也就是说它的雅各比阵 $\dfrac{\partial z_2}{\partial h_2}$ 是一个对角阵，具体形式大致是这样：

$$\frac{\partial z_2}{\partial h_2}\Big|_{h_2} = \begin{pmatrix} \phi'(\cdot) & & & \\ & \phi'(\cdot) & & \\ & & \ddots & \\ & & & \phi'(\cdot) \end{pmatrix}\Bigg|_{h_2} \tag{29}$$

这里表示将 h_2 的所有分量分别代入对角矩阵中的对应位置处。

因此综上所述，

$$\frac{\partial f}{\partial W_2} = \frac{\partial z_2}{\partial h_2}\left(\frac{\partial \hat{y}}{\partial z_2}\right)^\top \frac{\partial f}{\partial \hat{y}} z_1^\top \tag{30}$$

$$= \frac{\partial z_2}{\partial h_2} W_3^\top \frac{\partial f}{\partial \hat{y}} z_1^\top \tag{31}$$

进一步，$\dfrac{\partial z_2}{\partial h_2}$ 这一稀疏的对角矩阵似乎表达得不够紧凑，在计算机上实现效率不高。我们知道对角矩阵左乘相当于行变换，即每一行分别乘以对角矩阵的对应元素，而右边的 $W_3^\top \dfrac{\partial f}{\partial \hat{y}}$ 又恰好也是一个同一维度的列向量，因此我们可以将方阵与向量的乘积转化为两个向量的元素点积。我们记由 $\dfrac{\partial z_2}{\partial h_2}$ 的对角元构成的向量为 $\phi'(h_2)$，这一记号也是比较合理的，最终我们可以得到 W_2 导数在计算上比较高效的表达式：

$$\frac{\partial f}{\partial W_2} = \left[\phi'(h_2) \odot \left(W_3^\top \frac{\partial f}{\partial \hat{y}}\right)\right] z_1^\top \tag{32}$$

我们可以使用“维度检验法”来检验我们乘法的顺序是否正确，我们这里也做一演示。$\phi'(h_2) \in \mathbb{R}^{q\times 1}$, $W_3 \in \mathbb{R}^{s\times q}$, $\dfrac{\partial f}{\partial \hat{y}} \in \mathbb{R}^{s\times 1}$, $z_1 \in \mathbb{R}^{p\times 1}$，整体算式的维度为：

$$(q\times q)\cdot(q\times s)\cdot(s\times 1)\cdot(1\times p) \to q\times p \tag{33}$$

是正确的。

关于 W_1 的导数则是深度最深的一个导数，其计算过程为：

$$\frac{\partial f}{\partial W_1} = \frac{\partial f}{\partial h_1} x^\top \tag{34}$$

其中

$$\left(\frac{\partial f}{\partial h_1}\right)^\top = \left(\frac{\partial f}{\partial h_2}\right)^\top \frac{\partial h_2}{\partial z_1}\frac{\partial z_1}{\partial h_1} \tag{35}$$

$$= \left(\frac{\partial f}{\partial h_2}\right)^\top W_2 \frac{\partial z_1}{\partial h_1} \tag{36}$$

其中 $\dfrac{\partial f}{\partial h_2}$ 我们在计算 W_2 的导数时已经计算过，即 $\phi'(h_2)\odot\left(W_3^\top\dfrac{\partial f}{\partial \hat{y}}\right)$，这里可以作为已知条件。从而：

$$\frac{\partial f}{\partial W_1} = \frac{\partial z_1}{\partial h_1}W_2^\top\frac{\partial f}{\partial h_2}x^\top \tag{37}$$

$$= \left[\phi'(h_1)\odot\left(W_2^\top\frac{\partial f}{\partial h_2}\right)\right]x^\top \tag{38}$$

维度验证我们此处省略，同学们可以自己尝试。

3.4 总结

可以看出，反向传播是求梯度自然而然产生的现象，越靠近神经网络尾部的权重变量，其梯度依赖的信息就越少，而靠近头部的权重变量的梯度计算需要用到尾部的变量及参数的信息，因此整体求导的流程大致是一个从尾部向头部反向传播的过程。而且我们进一步可以发现，机器学习将梯度写作列向量是有好处的。在每一个梯度表达式中，除最后一个用引理表示的转换向量（z_2, z_1, x）外，其余向量的排布顺序基本和神经网络从左至右的排布顺序是相同的，方便我们十分直观地看出求导的反向传播过程。