目录

1 Backpropagation

1.1 Jacobian

现有映射：$f : R^N \to R^M$，即，接受一个向量 $x \in R^N$ 作为函数 $y = f(x)$ 的输入，返回一个向量 $y \in R^M$作为输出，这可以视作有M个函数，任意一个函数y_i接受N个输入，并返回一个标量作为输出。输出y对输入x的偏导数为雅可比矩阵(Jacobian)，

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_M}{\partial x_1} & \frac{\partial y_M}{\partial x_2} & \cdots & \frac{\partial y_M}{\partial x_N} \end{bmatrix}$$

其第i行第j列元素表示第i个函数y_i对第j个输入x_j的偏导数，那么，若$x \to x + \Delta x$，则 $y \to y + \frac{\partial y}{\partial x} \cdot \Delta x$ 其中·代表矩阵与向量之间的内积运算。矩阵有M行代表有M个函数，而N列则代表有N个输入，即，雅可比矩阵为一个$M \times N$的矩阵，恰好在点乘运算时的写法与通常的写法一致，即$\frac{\partial y}{\partial x} \cdot \Delta x$。

广义雅可比矩阵接受一个张量(Tensor)，输出一个张量，假设输入X与输出Y满足的维度分别为：D_X, D_Y，其形状分别为：$N_1 \times N_2 \times \cdots N_{D_X}fiM_1 \times M_2 \times \cdots M_{D_Y}$，则对应的广义雅可比矩阵的维度为：$D_X + D_Y$，形状为：$(M_1 \times M_2 \times \cdots M_{D_Y}) \times (N_1 \times N_2 \times \cdots N_{D_X})$，这里将形状归为两组，前者为输出张量Y的形状，后者为输入张量X的形状，与雅可比矩阵保持一致。

维度超过三维的张量通常无法直观理解，但是可以将其理解为一个高维数组，该数组可以看成一个向量或者一个矩阵，每一个元素都是一个高维数组，而最底层数据仍是标量。如：四维张量可以看作一个矩阵，矩阵的每一个元素都是子矩阵（也可以看作向量，其每一个元素又是一个向量），子矩阵的每一个元素都是一个标量。另一种观点为：将四维张量看作一个向量，向量的每一个元素都是一个三维的数组，该三维数组的每一个元素又可以看作矩阵、向量、标量。就像编程语言中索引数组的某一个元素一样，我们可以用由整数构成的向量$i \in Z^{D_X}, j \in Z^{D_Y}$来确定输入x与输出y的标量元素，且$(\frac{\partial y}{\partial x})_{j,i} = \frac{\partial y_j}{\partial x_i}$，表示由索引向量j确定的输出张量y的一个标量$y_j$对由索引向量i确定的输入张量X的一个标量$x_i$的变化率，且因为$x_i, y_j$都是标量，所以$(\frac{\partial y}{\partial x})_{j,i}$也是标量，可以由索引向量j和索引向量i联合确定其在广义雅可比矩阵中的位置。对广义雅可比矩阵，我们有：$x \to x + \Delta x, \Rightarrow y \to y + \frac{\partial y}{\partial x} \cdot \Delta x$，这里的点乘为广义的点乘，即，对于有索引向量i, j所确定的元素，有：$(\frac{\partial y}{\partial x})_i = \sum_j (\frac{\partial y}{\partial x})_{i,j} \cdot (\Delta x)_j = (\frac{\partial y}{\partial x})_{j,:} \cdot \Delta x$，也是对所有对应的元素进行相乘并求和，其中求和为对所有可能的索引向量j进行的，与矩阵、向量之间的定义相同。

实际上，对于高维张量的运算，我们接受输入形状与输出形状分别为：$N_1 \times N_2 \times \cdots N_{D_X}fiM_1 \times M_2 \times \cdots M_{D_Y}$ 的张量，仍然可以看成有 $M_1 \times M_2 \times \cdots M_{D_Y}$ 个函数，每一个函数接受 $N_1 \times N_2 \times \cdots N_{D_X}$ 个输入，这时，我们可以将输入展开成一个向量 $X \in R^{N_1 \times N_2 \times \cdots N_{D_X}}$，同时将输出展开成一个向量 $Y \in$

$M_1 \times M_2 \times \cdots M_{D_Y}$ 我们可以得到雅可比矩阵 $(\frac{\partial Y}{\partial X})_{(M_1 \times M_2 \times \cdots M_{D_Y}) \times (N_1 \times N_2 \times \cdots N_{D_X})}$，将矩阵元素按照对应关系放入高维张量中，便是广义雅可比矩阵。

1.2 Chain Rule

链式法则是复合函数求导的重要观点，是深度学习进行高效反向传播的基础。

对映射$f, g : R \to R$，假设有：$z = g(y), y = f(x)$，则很容易计算出：$\frac{\partial z}{\partial y}, \frac{\partial y}{\partial x}$，则根据链式法则：$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$

现在假设有映射：$f : R^N \to R^M, g : R^M \to R^K$，且有$z = g(y), y = f(x)$，则根据以上内容，易知$\frac{\partial z}{\partial y}$的形状为$(K) \times (M)$，$\frac{\partial y}{\partial x}$的形状为$(M) \times (N)$，$\frac{\partial z}{\partial x}$的形状为$(K) \times (N)$，根据链式法则：$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$恰好在形状上能够满足。实际上，若将X到Z的过程看作一个函数$z = h(x) = g[f(x)]$，以$\frac{\partial z}{\partial x}$的第i行第j列所确定的元素为例，它表示输入X的第j个元素对输出Z的第i个元素的作用，但实际上我们知道X是通过影响Y从而影响Z的，而Z_i是函数$Z = g(Y)$的第i个函数的输出，它受到Y中所有元素的影响，而Y又X的第j个输入有联系，我们可以很容易知道：$\frac{\partial z_i}{\partial x_j} = \sum_k \frac{\partial z_i}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_j}$，即$\frac{\partial z_i}{\partial x_j}$由$\frac{\partial z}{\partial y}$的第k行所确定的向量与$\frac{\partial y}{\partial x}$第j列所确定的向量所得内积。

对于张量而言，链式法则仍然成立，只是其中的点乘为广义点乘。

下面分别从仅含有标量梯度的链式法则、含有向量梯度的链式法则、含有矩阵梯度的链式法则进行推导。

1.2.1 仅含有标量梯度的链式法则

考虑两个映射：$f : R \to R, g : R \to R$，记：$y = f(x), z = g(y), z = g[f(x)] = h(x)$ ，即，$z = g[f(x)]$ 是一个复合函数，若要求 $\frac{\partial z}{\partial x}$，可以利用复合函数的求导法则。我们首先分别求出 z 对于 y 的梯度以及 y 对于 x 的梯度，即：$\frac{\partial z}{\partial y} = g'(y), \frac{\partial y}{\partial x} = f'(x)$，那么根据复合函数求导法则，我们有 $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} = g'(y) \cdot f'(x)$

1.2.2 含有向量梯度的链式法则

考虑两个映射：$f : R^N \to R, g : R \to R$，这里考虑的是神经网络的损失函数输出通常为一个标量，因此 g 为一个标量到标量的映射。记：$y = f(x), z = g(y), z = g[f(x)] = h(x)$ ，即，$z = g[f(x)]$ 是一个复合函数，若要求 $\frac{\partial z}{\partial x}$，我们依然可以利用复合函数的求导法则。同样，我们首先分别求出 z 对于 y 的梯度以及 y 对于 x 的梯度，前者将标量映射为标量，故其梯度仍然可以写为 $\frac{\partial z}{\partial y} = g'(y)$，而后者将一个 N 维向量映射为一个标量，根据多维函数求导法则，我们可以计算出输出 y 对每一个输入的导数 $\frac{\partial y}{\partial x} = [\frac{\partial y}{\partial x_1} \quad \frac{\partial y}{\partial x_2} \quad \cdots \quad \frac{\partial y}{\partial x_N}]$。由于 z 是关于 x 的复合函数，因此我们可以求出 z 关于 x 的梯度，即：

$$\begin{aligned} \frac{\partial z}{\partial x_1} &= \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_1} \\ \frac{\partial z}{\partial x_2} &= \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_2} \\ &\cdots \\ \frac{\partial z}{\partial x_N} &= \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_N} \end{aligned} \tag{1}$$

因此，我们可以将 z 关于 x 的梯度写为：$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$ 由于 $\frac{\partial z}{\partial y}$ 是标量而 $\frac{\partial y}{\partial x}$ 是向量，故 z 关于 x 的梯度为一个向量，且维度与 x 一致。

1.2.3 含有矩阵梯度的链式法则

考虑两个映射：$f : R^N \to R^M, g : R^M \to R$，记：$y = f(x), z = g(y), z = g[f(x)] = h(x)$ ，即，$z = g[f(x)]$ ，若要求 $\frac{\partial z}{\partial x}$，我们依然可以利用复合函数的求导法则。同样，我们首先分别求出 z 对于 y 的梯

度以及 y 对于 x 的梯度，前者将向量映射为标量，其梯度可以写为 $\frac{\partial z}{\partial y} = [\frac{\partial z}{\partial y_1} \quad \frac{\partial z}{\partial y_2} \quad \cdots \quad \frac{\partial z}{\partial y_M}]$，后者将一个 N 维向量映射为一个 M 维向量，故每一个输出 y_i 对每一个输入的导数 $\frac{\partial y_i}{\partial x} = [\frac{\partial y_i}{\partial x_1} \quad \frac{\partial y_i}{\partial x_2} \quad \cdots \quad \frac{\partial y_i}{\partial x_N}]$。由于 z 是关于 x 的复合函数，因此我们可以求出 z 关于 x 的梯度，即：

$$\begin{aligned}
\frac{\partial z}{\partial x_1} &= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1} + \cdots + \frac{\partial z}{\partial y_M} \cdot \frac{\partial y_M}{\partial x_1} \\
\frac{\partial z}{\partial x_2} &= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2} + \cdots + \frac{\partial z}{\partial y_M} \cdot \frac{\partial y_M}{\partial x_2} \\
&\cdots \\
\frac{\partial z}{\partial x_N} &= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_N} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_N} + \cdots + \frac{\partial z}{\partial y_M} \cdot \frac{\partial y_M}{\partial x_N}
\end{aligned} \tag{2}$$

将 $\frac{\partial y}{\partial x}$ 写成雅可比矩阵形式，有：

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \end{bmatrix} \tag{3}$$

那么我们可以将复合函数求导的展开式写成矩阵的表达形式，即：$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$

1.2.4 小结

至此，基本的链式法则形式我们已经推到完毕，需要注意的是，涉及到标量对行向量求导，结果仍然是行向量，涉及到向量对向量求导，结果为一个 $M \times N$ 的矩阵，其中，M 是输出的向量的维度，N 是输入的向量的维度。链式法则主要的思想便是基础的复合函数求导法，利用该求导法，我们可以方便地推导出神经网络中间各个层的参数的梯度。神经网络的输入输出往往是高维张量，这时候由于函数的所有输入与其中一个输出仍然由一个接受多个输入的函数确定，因此我们仍然可以将其看成一个向量的输入，产生一个向量输出，此时上方介绍的含矩阵梯度的链式法则仍然适用，具体可见1.1.

1.3 Backpropagation

反向传播算法是深度学习的基础，依赖于链式法则。对于一个由参数W确定的函数$y = f(x; W)$，我们可以求出$\frac{\partial y}{\partial W}$，然而对于n个函数嵌套而成的复合函数，如：$y = f_1(f_2((\cdots); W_2); W_1)$，想要求出$\frac{\partial y}{\partial W_1}$，最朴素的方法是推导出计算公式，然而这种方法很复杂，而且扩展性很不好，当中间有一个函数被其他函数替代时，往往需要重新推导。采用链式法则可以避免这种情况发生，即：$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$，其中，$\frac{\partial z}{\partial y}$称为upstream gradient，$\frac{\partial y}{\partial x}$称为local gradient，$\frac{\partial z}{\partial x}$称为downstream gradient，反向传播时，只需要接收提供的upstream gradient并计算出$y = f(x)$的local gradient，便能够通过链式法则求出$\frac{\partial z}{\partial x}$，当其他函数发生改变时，并不影响当前函数的梯度计算，因此各个函数所需要计算的梯度实际上是解耦的，只需要关心如何通过接收到的upstream gradient计算出downstream gradient即可，方便代码实现。