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$\cdot$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。

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。

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。