深度学习入门

从零构建CNN和RNN

Stephen CUI

March 17, 2023

Chapter 1

基本概念

1.1 函数

1.2 导数

1.3 嵌套函数

1.4 链式法则

1.5 示例介绍

数学

从数学上讲，对于包含 3 个基本可微的函数的复合函数，其导数的计算公式如下：

$$\frac{df_3}{du}(x) = \frac{df_3}{du}(f_2(f_1(x))) \times \frac{df_2}{du}(f_1(x)) \times \frac{df_1}{du}(x)$$

示意图

要理解以上公式，最为直观的方法就是通过盒子示意图，如 Figure 1.1 所示。

注意，在计算这个嵌套函数的链式法则时，这里对它进行了两次“传递”。

1. “向前”传递它，计算出 `f1_of_x` 和 `f2_of_x`，这个过程可以称作（或视作）“前向传递”。
2. “向后”通过函数，使用在前向传递中计算出的量来计算构成导数的量。最后，将这 3 个量相乘，得到导数。

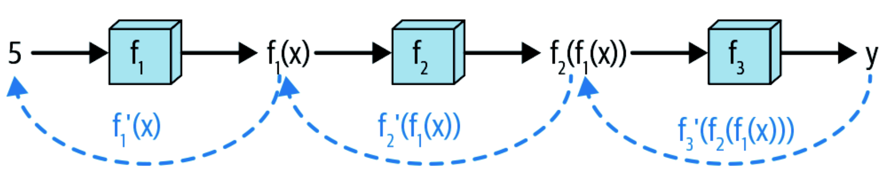

Figure 1.1: The box model for computing the derivative of three nested functions

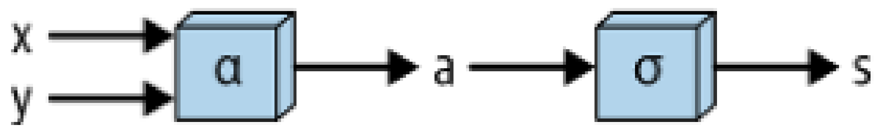

Figure 1.2: Function with multiple inputs

1.6 多输入函数

示意图

既然谈到了多输入函数，现在来定义我们一直在讨论的一个概念：用箭头表示数学运算顺序的示意图叫作**计算图**（computational graph）。

可以看到，两个输入进入 α 输出 a，然后 a 再被传递给了 σ 。

1.7 多输入函数的导数

1.8 多向量输入函数

数学

在神经网络中，表示单个数据点的典型方法是将 n 个特征列为一行，其中每个特征都只是一个数字，如 x_1 、x_2 等表示如下：

$$\boldsymbol{X} = [x_1, x_2, \cdots, x_n]$$

1.9 基于已有特征创建新特征

神经网络中最常见的运算也许就是计算已有特征的加权和，加权和可以强化某些特征而弱化其他特征，从而形成一种新特征，但它本身仅仅是旧特征的组合。用数学上的一种简洁的方式表达就是使用该观测值的点积（dot product），配合与特征 $w_1, w_2, \ldots, w_n$ 等长的一组权重。

数学

存在$\boldsymbol{W} = [w_1, w_2, \ldots, w_n]^T$，那么可以将此运算的输出定义为：

$$N = v(\boldsymbol{X}, \boldsymbol{W}) = \boldsymbol{X}\boldsymbol{W} = x_1w_1 + x_2w_2 + \cdots + x_nw_n$$

1.10 多向量输入函数的导数

如果将点积写为$v(\boldsymbol{X}, \boldsymbol{W}) = N$ 这种形式，那么自然会产生一个问题：$\frac{\partial N}{\partial \boldsymbol{X}}$和$\frac{\partial N}{\partial \boldsymbol{W}}$分别是什么？

数学

矩阵语法只是对一堆以特定形式排列的数字的简写，“矩阵的导数”实际上是指“矩阵中每个元素的导数”。由于 $\boldsymbol{X}$ 有一行，因此它可以这样定义：

$$\frac{\partial v}{\partial \boldsymbol{X}} = \left[\frac{\partial v}{\partial x_1} \frac{\partial v}{\partial x_2} \frac{\partial v}{\partial x_3}\right]$$

又有

$$N = v(\boldsymbol{X}, \boldsymbol{W}) = \boldsymbol{X}\boldsymbol{W} = x_1w_1 + x_2w_2 + \cdots + x_nw_n$$

因此可以得出：

$$\frac{\partial v}{\partial x_1}=w_1,\frac{\partial v}{\partial x_2}=w_2,\frac{\partial v}{\partial x_3}=w_3$$
$$\frac{\partial v}{\partial \boldsymbol{X}}=[w_1,w_2,w_3]=\boldsymbol{W}^T$$

这个结果出乎意料地简练，掌握这一点极为关键，既可以理解深度学习的有效性，又可以知道如何清晰地实现深度学习。

以此类推，可以得到如下公式：

$$\frac{\partial v}{\partial \boldsymbol{W}}=\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\boldsymbol{X}^T$$

代码

这里计算的 `dNdX` 表示 $\boldsymbol{X}$ 的每个元素相对于输出 N 的和的偏导数。在本书中，这个量有一个特殊的名称，即 $\boldsymbol{X}$ 的梯度（gradient）。这个概念是指，对于 $\boldsymbol{X}$ 的单个元素（例如 x_3 ）， `dNdX` 中的对应元素（具体来说是 `dNdX[2]`）是向量点积 N 的输出相对于 $x+3$ 的偏导数。

1.11 向量函数及其导数：再进一步

假设函数接受向量$\boldsymbol{X}$ 和向量 $\boldsymbol{W}$，执行点积（将其表示为 $v(\boldsymbol{X},\boldsymbol{W})$，然后将向量输入到函数$\sigma$中。

数学

公式很简单，如下所示：

$$s=f(\boldsymbol{X},\boldsymbol{W})=\sigma(v(\boldsymbol{X},\boldsymbol{W}))=\sigma(x_1w_1+x_2w_2+x_3w_3)$$

向量函数及其导数：后向传递

数学

由于$f(\boldsymbol{X},\boldsymbol{W})=\sigma(v(\boldsymbol{X},\boldsymbol{W}))$，因此该函数在$\boldsymbol{X}$出的导数可以这样表示：

$$\begin{aligned}\frac{\partial f}{\partial \boldsymbol{X}}&=\frac{\partial \sigma}{\partial u}(v(\boldsymbol{X},\boldsymbol{W}))\frac{\partial v}{\partial \boldsymbol{X}}(\boldsymbol{X},\boldsymbol{W})\\&=\frac{\partial \sigma}{\partial u}(x_1w_1+x_2w_2+x_3w_3)\boldsymbol{W}^T\end{aligned}$$

1.12 包含两个二维矩阵输入的计算图（更一般、更实际的情况）

在深度学习和更通用的机器学习中，需要处理输入为两个二维数组的运算，其中一个数组表示一批数据 $\boldsymbol{X}$，另一个表示权重 $\boldsymbol{W}$。

数学

假设$\boldsymbol{X}$和$\boldsymbol{W}$如下所示：

$$\boldsymbol{X}=\begin{bmatrix}x_{11}&x_{12}&x_{13}\\x_{21}&x_{22}&x_{23}\\x_{31}&x_{32}&x_{33}\end{bmatrix}$$
$$\boldsymbol{W}=\begin{bmatrix}w_{11}&w_{12}\\w_{21}&w_{22}\\w_{31}&w_{32}\end{bmatrix}$$

这可能对应一个数据集，其中每个观测值都具有 3 个特征，即矩阵的列数对应特征数，3 行可能对应要对其进行预测的 3 个观测值，即矩阵的行数对应观测值。

现在将为这些矩阵定义以下简单的运算。

1. 将这些矩阵相乘。和以前一样，将把执行此运算的函数表示为 $v(\boldsymbol{X}, \boldsymbol{W})$，将输出表示为$N$。
2. 将结果N 传递给可微函数 σ ，并定义$S = \sigma(N)$。

现在的问题是：输出 S 相对于 X 和 W 的梯度是多少？

这就引出了一个微妙但十分重要的概念：可以在目标多维数组上执行任何一系列运算，但是要对某些输出定义好梯度，这需要对序列中的最后一个数组求和（或以其他方式聚合成单个数字），这样“X 中每个元素的变化会在多大程度上影响输出”这一问题才有意义。

$$\begin{aligned}\boldsymbol{XW} &= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{21} + x_{13}w_{31} & x_{11}w_{12} + x_{12}w_{22} + x_{13}w_{32} \\ x_{21}w_{11} + x_{22}w_{21} + x_{23}w_{31} & x_{21}w_{12} + x_{22}w_{22} + x_{23}w_{32} \\ x_{31}w_{11} + x_{32}w_{21} + x_{33}w_{31} & x_{31}w_{12} + x_{32}w_{22} + x_{33}w_{32} \end{bmatrix} \\ &= \begin{bmatrix} XW_{11} & XW_{12} \\ XW_{21} & XW_{22} \\ XW_{31} & XW_{32} \end{bmatrix}\end{aligned}$$

为了便于书写结果矩阵，这里将第 i 行的第 j 列表示为 XW_{ij}。

接下来，将该结果输入到 σ 中：

$$\sigma(\boldsymbol{XW}) = \begin{bmatrix} \sigma(XW_{11}) & \sigma(XW_{12}) \\ \sigma(XW_{21}) & \sigma(XW_{22}) \\ \sigma(XW_{31}) & \sigma(XW_{32}) \end{bmatrix}$$

最后，对这些元素求和：

$$\begin{aligned} L = \Lambda(\sigma(\boldsymbol{XW})) = \Lambda\left(\begin{bmatrix} \sigma(XW_{11}) & \sigma(XW_{12}) \\ \sigma(XW_{21}) & \sigma(XW_{22}) \\ \sigma(XW_{31}) & \sigma(XW_{32}) \end{bmatrix}\right) \\ = \sigma(XW_{11}) + \sigma(XW_{12}) + \sigma(XW_{21}) + \sigma(XW_{22}) + \sigma(XW_{31}) + \sigma(XW_{32}) \end{aligned}$$

现在回到了纯微积分的场景中：存在一个数字 L，想计算出 L 相对于 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 的梯度，也就是明确这些输入矩阵中每个元素的变化对 L 的影响。可以这样写：

$$\frac{\partial \Lambda}{\partial u} = \begin{bmatrix} \frac{\partial \Lambda}{\partial u}(x_{11}) & \frac{\partial \Lambda}{\partial u}(x_{12}) & \frac{\partial \Lambda}{\partial u}(x_{13}) \\ \frac{\partial \Lambda}{\partial u}(x_{21}) & \frac{\partial \Lambda}{\partial u}(x_{22}) & \frac{\partial \Lambda}{\partial u}(x_{23}) \\ \frac{\partial \Lambda}{\partial u}(x_{31}) & \frac{\partial \Lambda}{\partial u}(x_{32}) & \frac{\partial \Lambda}{\partial u}(x_{33}) \end{bmatrix}$$

1.13 有趣的部分：向后传递

数学

值 L 实际上是$x_{11}, x_{12}, x_{13}, x_{21}, \cdots, x_{33}$的一个函数。

矩阵显著地分解成：

$$\frac{\partial \Lambda}{\partial u}(\boldsymbol{X}) = \frac{\partial \Lambda}{\partial u}(S) \times \frac{\partial \sigma}{\partial u}(N) \boldsymbol{X}^T$$

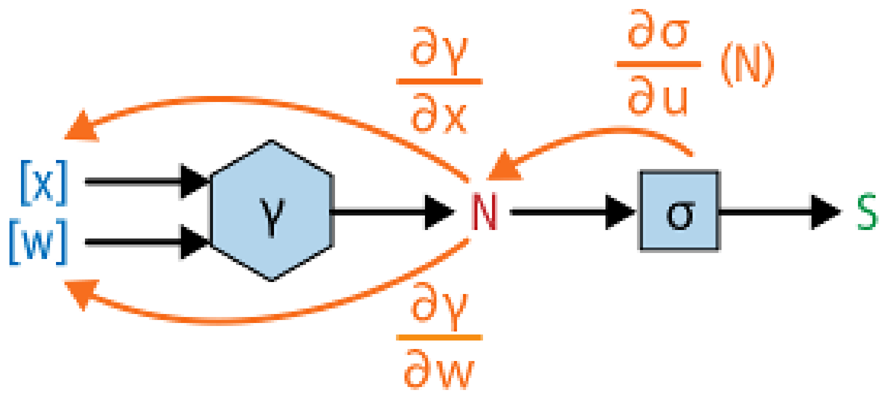

Figure 1.3: Graph with a matrix multiplication-the backward pass

其中前两个是逐个元素执行的，第三个则是矩阵乘法。

L 相对于 $\boldsymbol{W}$ 的梯度的表达式为 $\boldsymbol{X}^T$ 。但是，$\boldsymbol{X}^T$ 表达式中的因子是从 L 的导数中导出的，考虑到它们的顺序，$\boldsymbol{X}^T$ 将位于 L 相对于 $\boldsymbol{W}$ 的梯度的表达式的左侧：

$$\frac{\partial \Lambda}{\partial u}(\boldsymbol{W}) = \boldsymbol{X}^T \frac{\partial \Lambda}{\partial u}(S) \times \frac{\partial \sigma}{\partial u}(N)$$