

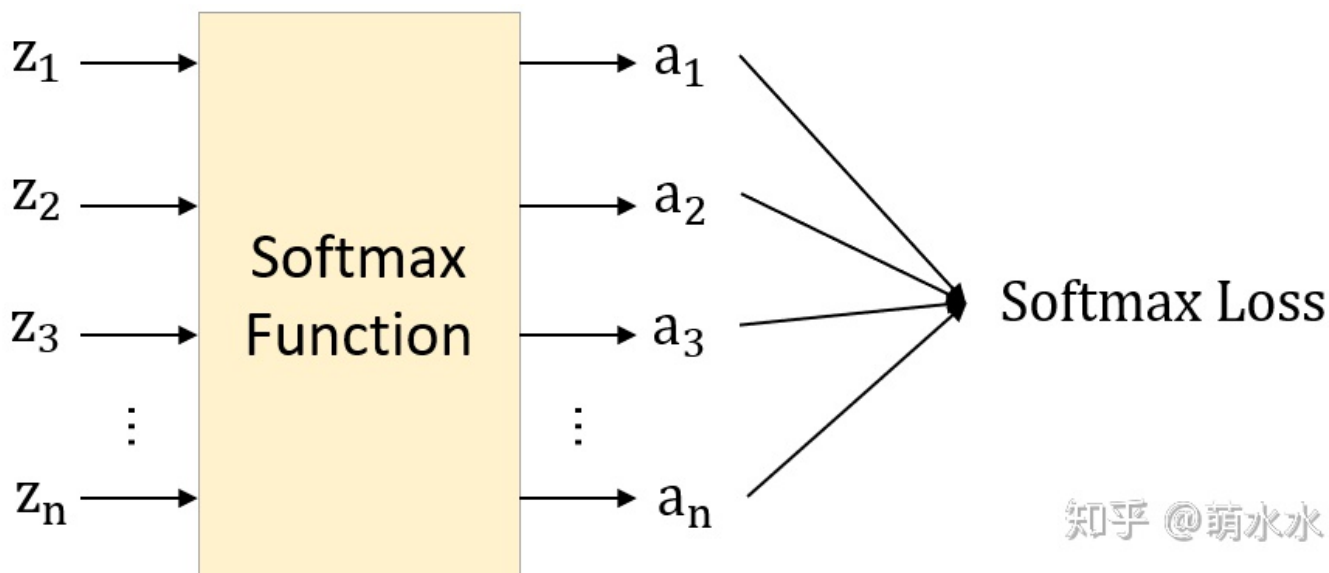
15分钟搞定Softmax Loss求导



萌水水 
只爱干货的萌水水

75 人赞同了该文章

求softmax loss过程如下图所示：



Softmax Loss 计算过程

符号定义如下：

1. 输入为 \mathbf{z} 向量， $\mathbf{z} = [z_1, z_2, z_3, \dots, z_n]$ ，维度为 $(1, n)$

2. 经过softmax函数， $a_i = \frac{e^{z_i}}{\sum_{k=1}^n e^{z_k}}$



知乎 @萌水水

3. Softmax Loss损失函数定义为 L , $L = -\sum_{i=1}^n y_i \ln(a_i)$, L 是一个标量, 维度为 (1,1)

其中 y 向量为模型的Label, 维度也是 (1, n), 为已知量, 一般为onehot形式。

我们假设第 j 个类别是正确的, 则 $y = [0, 0, \dots, 1, \dots, 0]$, 只有 $y_j = 1$, 其余 $y_i = 0$

那么 $L = -y_j \ln(a_j) = -\ln(a_j)$

我们的**目标**是求 标量 L 对向量 z 的导数 $\frac{\partial L}{\partial z}$ 。

由链式法则, $\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} * \frac{\partial a}{\partial z}$, 其中 a 和 z 均为维度为 (1, n) 的向量。

L 为标量, 它对向量 a 求导。标量对向量求导, 见CS224N Lecture 3的一页ppt:

Gradients

- Given a function with 1 output and n inputs

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$$

- It's gradient is a vector of partial derivatives with respect to each input

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right]$$

知乎 @萌水水

可知, 标量对向量求导, 维度不变, 也即 $\frac{\partial L}{\partial z}$ 和 $\frac{\partial L}{\partial a}$ 维度都应为 (1, n)。



Jacobian Matrix: Generalization of the Gradient

- Given a function with **m outputs** and n inputs

$$\mathbf{f}(\mathbf{x}) = [f_1(x_1, x_2, \dots, x_n), \dots, f_m(x_1, x_2, \dots, x_n)]$$

- It's Jacobian is an **$m \times n$ matrix** of partial derivatives

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_{ij} = \frac{\partial f_i}{\partial x_j}$$

知乎 @萌水水

其中 \mathbf{f} 对应的就是softmax函数， \mathbf{x} 对应的为 \mathbf{z} 。可知向量 \mathbf{a} 对向量 \mathbf{z} 求导是一个Jacobian矩阵，维度为 (n, n) 。

$\frac{\partial L}{\partial \mathbf{a}}$ 维度为 $(1, n)$ ， $\frac{\partial \mathbf{a}}{\partial \mathbf{z}}$ 维度为 (n, n) ， $\frac{\partial L}{\partial \mathbf{z}} = \frac{\partial L}{\partial \mathbf{a}} * \frac{\partial \mathbf{a}}{\partial \mathbf{z}}$ 。由矩阵乘法，维度 $(1, n) * (n, n) = (1, n)$ 。 $\frac{\partial L}{\partial \mathbf{z}}$ 的维度为 $(1, n)$ ，这个是合理的。

1. 求 $\frac{\partial L}{\partial \mathbf{a}}$

由 $L = -y_j \ln(a_j) = -\ln(a_j)$ ，可知最终的Loss只跟 a_j 有关。

$$\frac{\partial L}{\partial \mathbf{a}} = [0, 0, 0, \dots, -\frac{1}{a_j}, \dots, 0]$$

2. 求 $\frac{\partial \mathbf{a}}{\partial \mathbf{z}}$

\mathbf{a} 是一个向量， \mathbf{z} 也是一个向量，则 $\frac{\partial \mathbf{a}}{\partial \mathbf{z}}$ 是一个Jacobian矩阵，类似这样：



$$\begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \frac{\partial a_1}{\partial z_2} & \dots & \frac{\partial a_1}{\partial z_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial a_n}{\partial z_1} & \frac{\partial a_n}{\partial z_2} & \dots & \frac{\partial a_n}{\partial z_n} \end{bmatrix}$$

可以发现其实Jacobian矩阵的每一行对应着 $\frac{\partial a_i}{\partial \mathbf{z}}$ 。

由于 $\frac{\partial L}{\partial \mathbf{a}}$ 只有第j列不为0，由矩阵乘法，其实我们只要求 $\frac{\partial \mathbf{a}}{\partial \mathbf{z}}$ 的第j行，也即 $\frac{\partial a_j}{\partial \mathbf{z}}$ ，
 $\frac{\partial L}{\partial \mathbf{z}} = -\frac{1}{a_j} * \frac{\partial a_j}{\partial \mathbf{z}}$ ，其中 $a_j = \frac{e^{z_j}}{\sum_{k=1}^n e^{z_k}}$ 。

(1) 当 $i \neq j$

$$\frac{\partial a_j}{\partial z_i} = \frac{0 - e^{z_j} e^{z_i}}{(\sum_k^n e^{z_k})^2} = -a_j a_i$$

$$\frac{\partial L}{\partial z_i} = -a_j a_i * -\frac{1}{a_j} = a_i$$

(2) 当 $i = j$

$$\frac{\partial a_j}{\partial z_j} = \frac{e^{z_j} \sum_k^n e^{z_k} - e^{z_j} e^{z_j}}{(\sum_k^n e^{z_k})^2} = a_j - a_j^2$$

$$\frac{\partial L}{\partial z_j} = (a_j - a_j^2) * -\frac{1}{a_j} = a_j - 1$$

所以， $\frac{\partial L}{\partial \mathbf{z}} = [a_1, a_2, \dots, a_j - 1, \dots, a_n] = \mathbf{a} - \mathbf{y}$ 。

Softmax Cross Entropy Loss的求导结果非常优雅，就等于预测值与Label的差。

下面这段话挺好：

使用交叉熵误差作为softmax 函数的损失函数后，反向传播得到 $(y_1 - t_1, y_2 - t_2, y_3 - t_3)$ 这样“漂亮”的结果。实际上，这样“漂亮”的结果并不是偶然的，而是为了得到这样的结果，特意设计了交叉熵误差函数。回归问题中输出层使用“恒等函数”，损失函数使用“平方和误差”，也是出于同样的理由（3.5 节）。也就是说，使用“平方和误差”作为“恒等函数”的损失函数，反向传播才能得到 $(y_1 - t_1, y_2 - t_2, y_3 - t_3)$ 这样“漂亮”的结果。



引用:

1. Stanford CS224N web.stanford.edu/class/...
2. 深度学习入门：基于Python的理论与实现

编辑于 2020-02-26

写下你的评论...



一条咸鱼

2020-02-25

loss是不是少了一个负号

👍 1



萌水水 (作者) 回复 一条咸鱼

2020-02-26

说的对，已全部修改。

👍 赞



GraphSAGE

07-17

预测值与Label的差，这个真好记，和LR形势类似，不过LR多一个x

👍 赞



阿乐哉

07-01

看了那么多材料，就这一篇讲明白了

👍 赞



骰子

06-28

绝绝子！给大佬点个赞！

👍 赞



范虚

02-19

感谢，讲得非常清晰

👍 赞



子光

01-16

非常好的文章，解答了我困惑许久的问题



