# 目录



# 1 Activation

## 1.1 Sigmoid

Sigmoid 函数定义如下：$\sigma(x) = \frac{1}{1+e^{-x}}$，则其导函数为：$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \sigma(x) \cdot (1 - \sigma(x))$，Sigmoid 函数的图形和其导数的图形如图1所示：

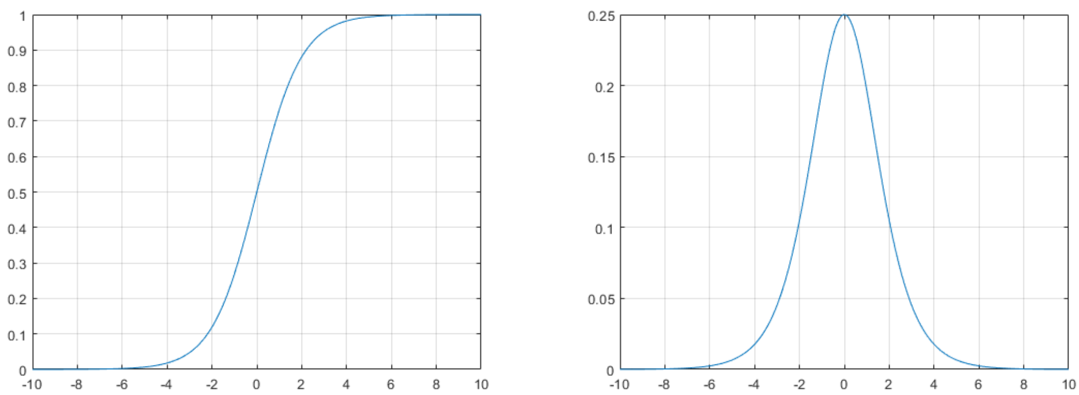

图 1: 左图为 Sigmoid 函数图像，右图为 Sigmoid 函数的导函数的图像

该函数将输入从$[-\infty, \infty]$映射到$[0, 1]$，但是它存在着3个问题：

- 其饱和特性会抑制神经元的梯度；
- 其输出并不以0为中心，假设神经元的输入总是为正(Sigmoid的输出本身就总是为正)，那么其权重的梯度便总是与upstream gradient的符号相同；
- 指数运算代价大；

现在我们考察 Sigmoid 函数在反向传播过程中的特性。考虑一个全连接神经网络的中间某一层隐藏层，其激活函数为 Sigmoid 函数，反向传播过程中，该隐藏层接受 upstream gradient $g_{up}$，梯度经过 sigmoid 函数后，其梯度变为 $g_a = g_{up} \odot sigmoid'(x)$，其中 $\odot$ 表示按元素相乘；而权重参数的梯度为 $g_W = X^T \cdot g_a$，由图 1 可知，激活函数的输出趋近于1或者趋近于0，则其梯度趋近于0，相应地，该隐藏层的权重梯度趋近于0。因此，在整个网络中，sigmoid 激活函数带来的权重衰减效应会一直累积，形成指数衰减，因此越靠近输入的隐藏层的梯度越接近0，这就导致梯度消失现象。

## 1.2 tanh

tanh 函数定义如下：$tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$，其导函数为：$tanh'(x)=\frac{(e^x+e^{-x})^2-(e^x-e^{-x})^2}{(e^x+e^{-x})^2}=1-tanh^2(x)$。tanh 函数与其相应的导函数的图像如图2所示：

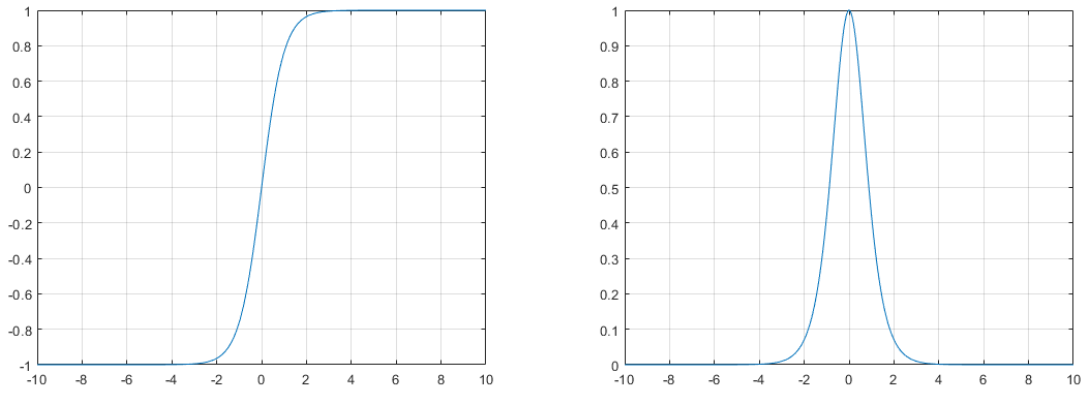


图 2: 左图为 tanh 函数图像，右图为 tanh 函数的导函数的图像

该函数将输入从$[-\infty,\infty]$映射到$[0,1]$，虽然输出以零为中心，但是仍然有者饱和的问题。

现在我们考察 tanh 函数在反向传播过程中的特性。考虑一个全连接神经网络的中间某一层隐藏层，其激活函数为 tanh 函数，反向传播过程中，基于与 sigmoid 函数同样的理由并且由图 1 可知，激活函数的输出趋近于1或者趋近于0，则其梯度趋近于0，相应地，该隐藏层的权重梯度趋近于0。因此，在整个网络中，tanh 函数与 sigmoid 激活函数一样，也会带来的权重衰减效应，且也会会一直累积，形成指数衰减，因此越靠近输入的隐藏层的梯度越接近0，这就导致 tanh 函数也会出现梯度消失现象。

## 1.3 ReLU: Recified Linear Unit

ReLU 函数及其导函数的定义如下：

$$ReLU(x)=\begin{cases}0 & ifx\leq 0\\ x & ifx\geq 0\end{cases}\quad ReLU'(x)=\begin{cases}0 & ifx\leq 0\\ 1 & ifx\geq 0\end{cases}\tag{1}$$

ReLU 函数及其对应的导函数图像如图3所示：

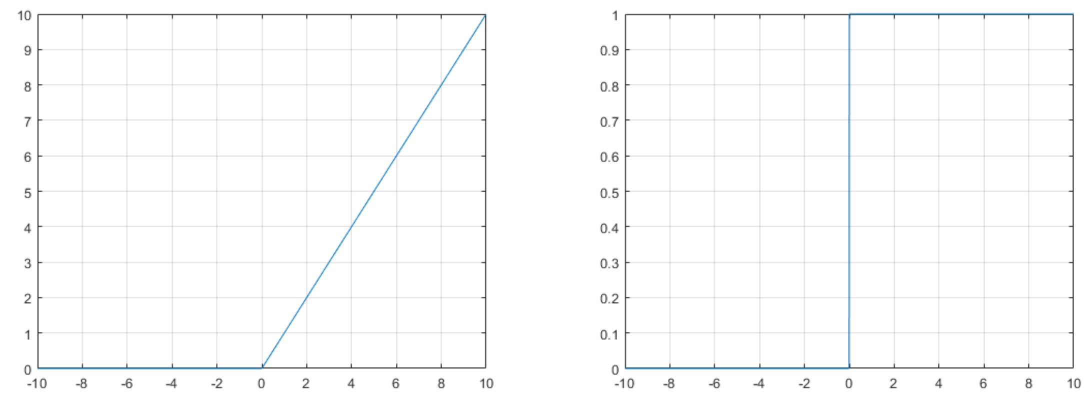


图 3: 左图为 ReLU 函数图像，右图为 ReLU 函数的导函数的图像

该函数不会饱和，并且计算高效，实际使用中收敛速度明显快于sigmoid与tanh，但它的输出并不以0为中心。相比于 sigmoid 函数和 tanh 函数，其反向传播过程中，对于正输入，其梯度不会发生变化，因此对于这部分输入，ReLU 函数不会产生梯度消失现象。然而 ReLU 函数也存在问题，对于输入为负的部分，其梯度相当于被直接置零，因此这一部分对应的参数将不会被更新，为了解决这个问题，通常使用 ReLU 函数时，相应的输出偏置应设置为一个略大于0的数。此外，有不少关于 ReLU 的变体也能够用于解决这一问题，如：Leaky ReLU, PReLU, ELU, SELU, Maxout 等，简要介绍如下。

## 1.4 Leaky ReLU

Leaky ReLU = max(0.01x, x)，该函数有着ReLU的优点，且反向传播时梯度不会直接变成0，其函数图像如图 4 所示。若将0.01设置成其他的参数，便是PReLU = $\max(\alpha x, x)$。对于 PReLU，其导函数如下：

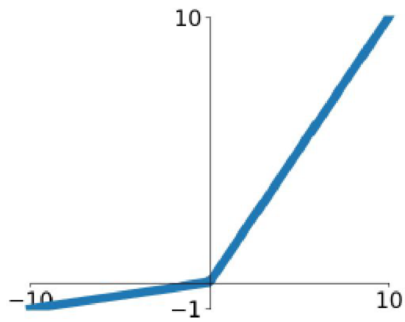


图 4: Leaky ReLU 的函数图像

$$PReLU'(x) = \begin{cases} \alpha & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{2}$$

对于 Leaky ReLU，其导函数相当于 $\alpha = 0.01$ 时的 PReLU。

## 1.5 ELU: Exponential Linear Units

ELU 函数定义如下：

$$f(x) = \begin{cases} x, x > 0 \\ \alpha(e^x - 1), x \leq 0 \end{cases}$$

其函数图像如图 5 所示。

该函数与Leaky ReLU相比，在负饱和的地方增加了对噪声的鲁棒性，但是需要指数计算，因此计算效率低。

## 1.6 SELU

SELU 函数的定义下：

$$f(x) = \begin{cases} \lambda x, x > 0 \\ \lambda\alpha(e^x - 1), x \leq 0 \end{cases}$$

其函数图像如图 6 所示。

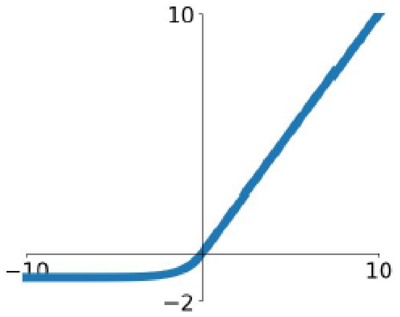


图 5: ELU 的函数图像

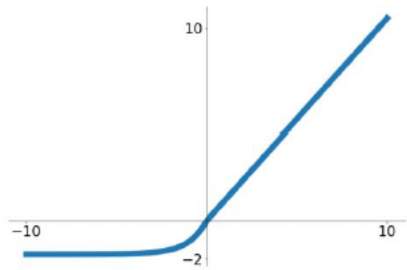


图 6: SELU 的函数图像

在深度学习中效果更好，具有自归一化的作用。

## 1.7 Maxout

out $= \max(w_1^T x + b_1, w_2^T x + b_2)$，ReLU和Leaky ReLU 等函数相当于是其特例，不饱和也不陷入死区，但参数增加了。实际中，通常应该先使用ReLU，然后尝试更换成 Leaky ReLU / Maxout / ELU / SELU 等来获得小幅度的性能提升，应该避免使用sigmoid和tanh。