智能之门

神经网络和深度学习入门

（基于Python的实现）

STEP 4 非线性回归

第 8 章

激活函数

在两层神经网络之间，必须有激活函数连接，从而加入非线性因素，提高神经网络的能力。所以，我们先从激活函数学起，一类是挤压型的激活函数，常用于简单网络的学习；另一类是半线性的激活函数，常用于深度网络的学习。

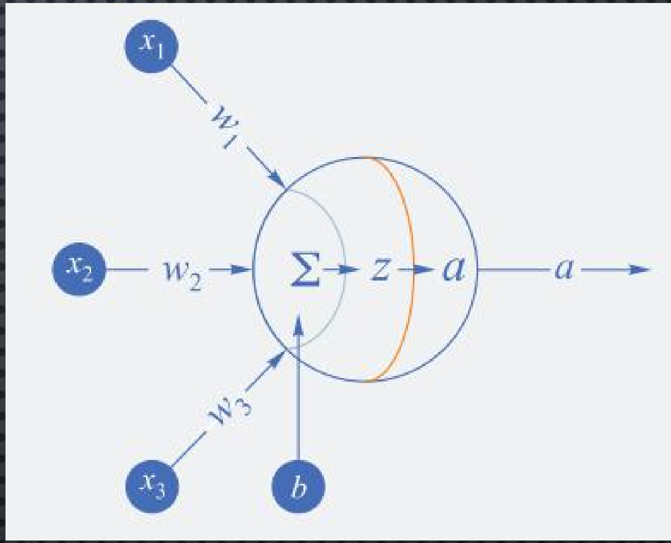
8.1 激活函数概论

右图是神经网络中的一个神经元，假设该神经元有三个输入单元，分别为 x_1, x_2, x_3，那么：

$$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$a = \sigma(z)$$

第二步也就是激活函数了，它的作用是给神经网络增加非线性因素，并将中间结果归一化，方便后面的运算。



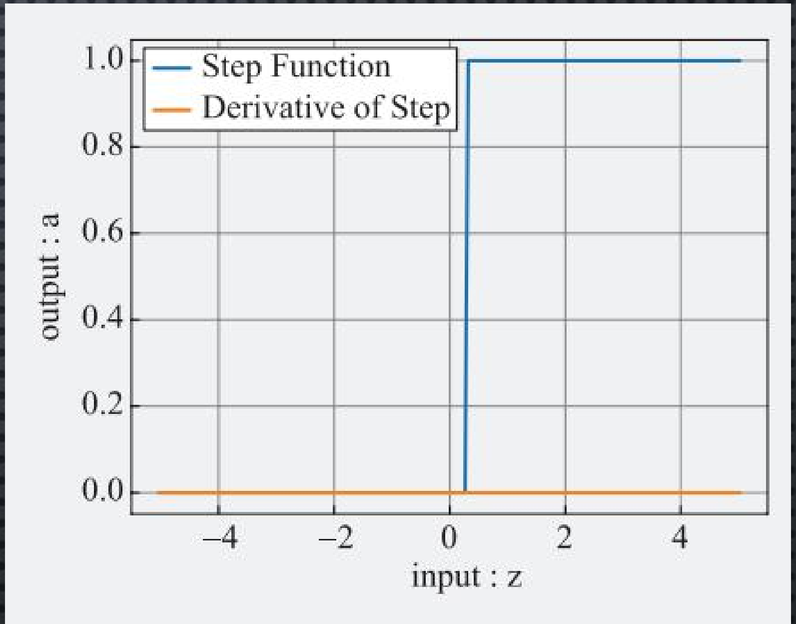
➢ 激活函数的性质

- 非线性：采用线性激活函数的作用为零；
- 可导性：做误差反向传播和梯度下降，必须要保证激活函数的可导性；
- 单调性：单一的输入会得到单一的输出，较大值的输入得到较大值的输出。

8.1 激活函数概论

在物理试验中使用的继电器，是最初激活函数的原型：当输入电流大于一个阈值时，会产生足够的磁场，从而打开下一级电源通道，如右图所示。用到神经网络中的概念，用1来代表一个神经元被激活，0代表一个神经元未被激活。

这个阶跃函数的缺点是，它的梯度（导数）恒为零（个别点除外)。反向传播公式中，梯度传递用到了链式法则，如果在这样一个连乘的式子其中有一项是零，总梯度就会恒为零，这意味着神经网络无法进行反向传播。



8.1 激活函数概论

激活函数用在神经网络层与层之间的连接，最后一层不用激活函数。在单层神经网络中，我们学习了以下内容：

输入	输出	激活函数	分类函数	功能
单变量	单输出	无	无	线性回归
多变量	单输出	无	无	线性回归
多变量	单输出	无	二分类函数	二分类
多变量	多输出	无	多分类函数	多分类

从上表可以看到，我们一直没有使用激活函数，而只使用了分类函数。对于多层神经网络也是如此，在最后一层只会用到分类函数来完成二分类或多分类任务，如果是拟合任务，则不需要分类函数。以后当不需要指定具体的激活函数形式时，会使用 σ 来代表激活函数。

8.2 挤压型激活函数

这一类函数的特点是，当输入值域的绝对值较大的时候，其输出在两端是饱和的，都具有S形的函数曲线以及压缩输入值域的作用，所以叫挤压型激活函数，又可以叫饱和型激活函数。

在英文中，通常用 $Sigmoid$ 来表示，原意是S型的曲线，在数学中是指一类具有压缩作用的S型的函数，在神经网络中，有两个常用的 $Sigmoid$ 函数，一个是 $Logistic$ 函数，另一个是 $Tanh$ 函数。

➢ 本系列中的约定

- $Sigmoid$： 指对数几率函数用于激活函数时的称呼；
- $Logistic$： 指对数几率函数用于二分类函数时的称呼；
- $Tanh$： 指双曲正切函数用于激活函数时的称呼。

8.2 挤压型激活函数

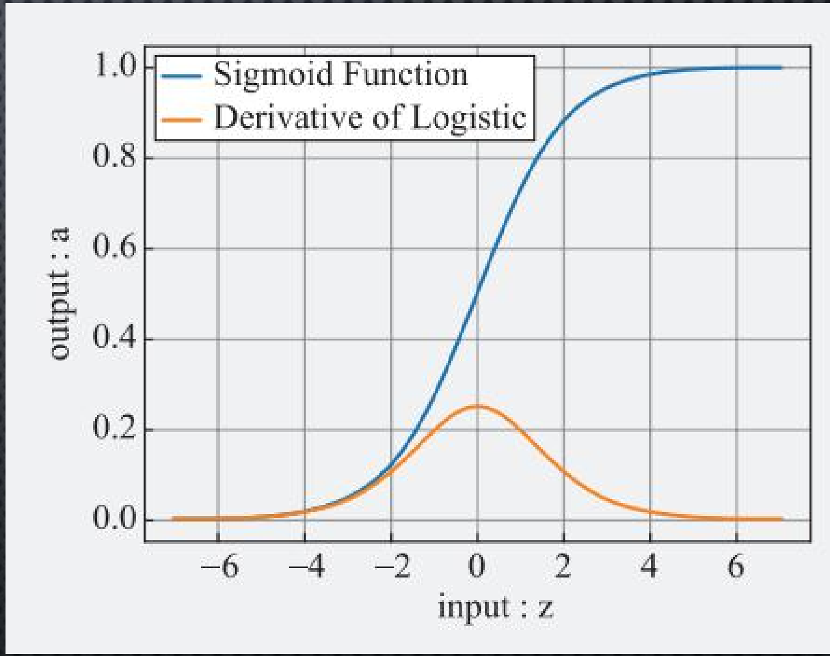
➢ ***Logistic* 函数**

- 函数表达式和导数公式

$$a = Logistic(z) = \frac{1}{1+e^{-z}}$$

$$Logistic'(z) = a(1-a)$$

- 定义域和值域
  - ✓ 输入定义域：$(-\infty, \infty)$；
  - ✓ 函数输出域：$(0,1)$；
  - ✓ 导函数输出域：$(0,0.25]$。



8.2 挤压型激活函数

- 优点
  - ✓ 从函数图像来看， $Sigmoid$ 函数的作用是将输入压缩到 $(0,1)$ 这个区间范围内，这种输出在 $(0,1)$ 间的函数可以用来模拟一些概率分布的情况，还是一个连续函数，导数简单易求。
  - ✓ 从数学上来看， $Sigmoid$ 函数对中央区的信号增益较大，对两侧区的信号增益小，在信号的特征空间映射上，有很好的效果。
  - ✓ 从神经科学上来看，中央区酷似神经元的兴奋态，两侧区酷似神经元的抑制态，因而在神经网络学习方面，可以将重点特征推向中央区， 将非重点特征推向两侧区。
- 缺点
  - ✓ 指数计算代价大。
  - ✓ 反向传播时梯度消失。

8.2 挤压型激活函数

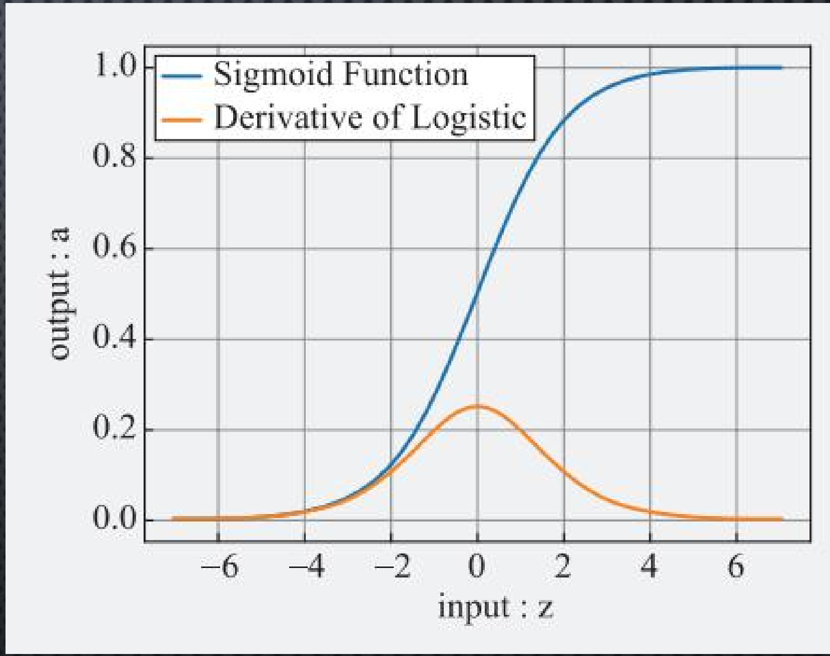
➢ *Tanh* 函数

- 函数表达式和导数公式

$$a = Tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2 \cdot Sigmoid(2z) - 1$$

$$Tanh'(z) = (1 + a)(1 - a)$$

- 定义域和值域
  - ✓ 输入定义域：$(-\infty, \infty)$；
  - ✓ 函数输出域：$(-1,1)$；
  - ✓ 导函数输出域：$(0,1]$。

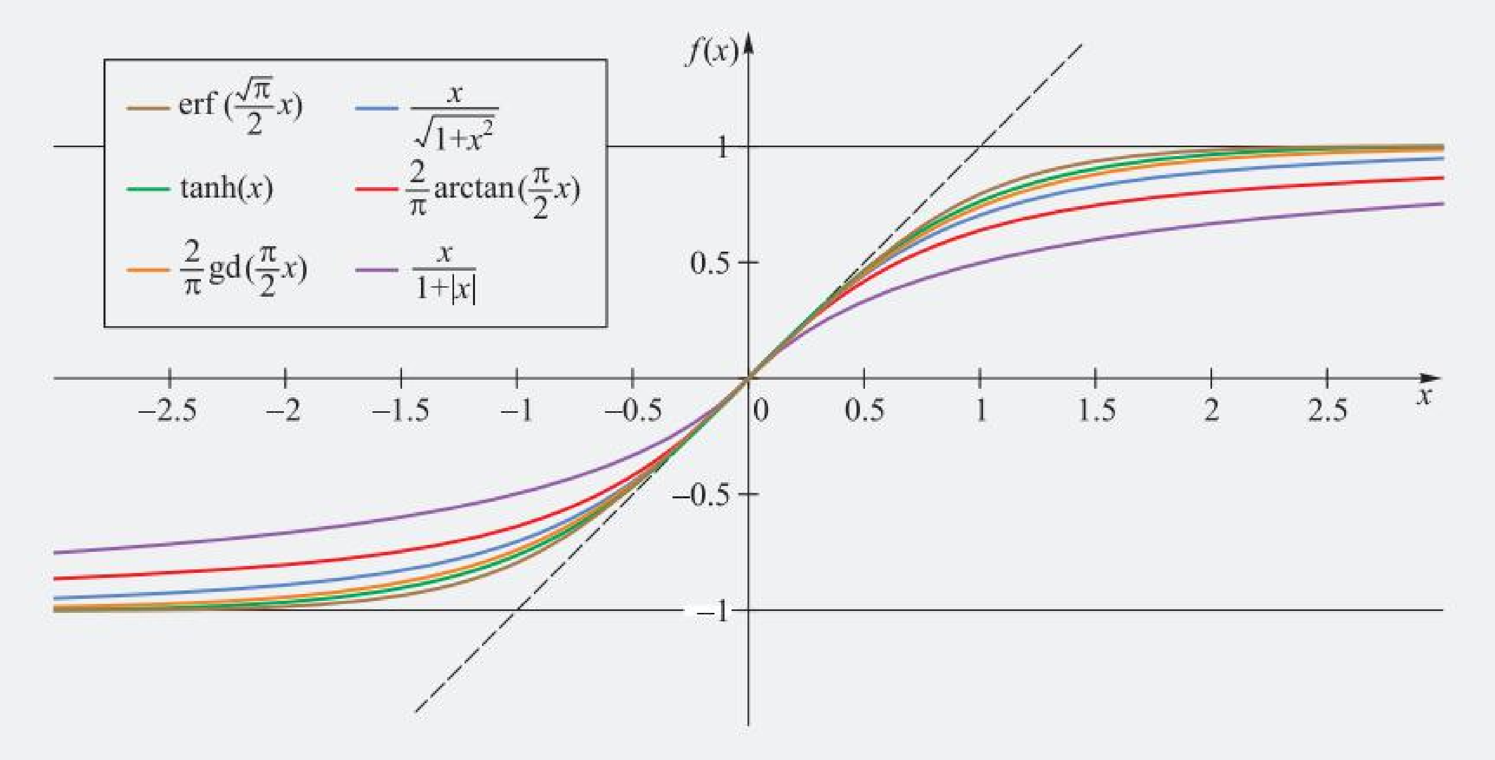


8.2 挤压型激活函数

- 优点
  - ✓ 具有前述 $Logistic$ 函数的所有优点。
  - ✓ 零均值，也就是说在传递过程中，输入数据的均值并不会发生改变。
- 缺点
  - ✓ 指数计算代价大。
  - ✓ 反向传播时梯度消失。

8.2 挤压型激活函数

➢ 其它函数



8.3 半线性激活函数

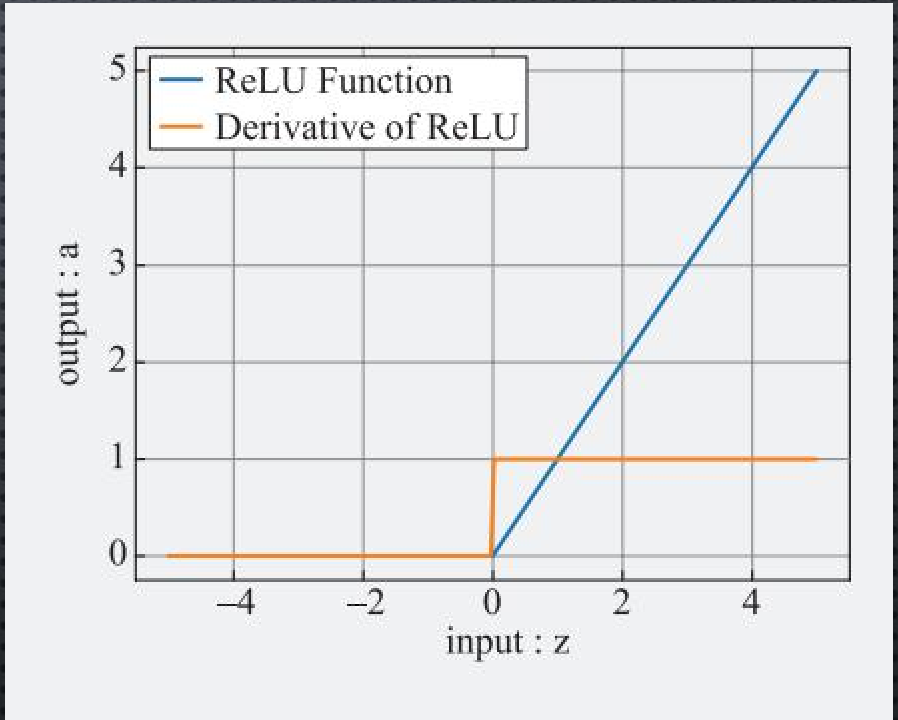
➢ *ReLU* 函数

- 函数表达式和导数公式

$$ReLU(z) = max(0, z)$$

$$ReLU'(z) = \begin{cases} 1, z \geq 0 \\ 0, z < 0 \end{cases}$$

- 定义域和值域
  - ✓ 输入定义域：$(-\infty, \infty)$；
  - ✓ 函数输出域：$[0, \infty)$；
  - ✓ 导函数输出域：$\{0,1\}$。



8.3 半线性激活函数

- 优点
  - ✓ 反向导数恒等于 1，更加有效率的反向传播梯度值，收敛速度快；
  - ✓ 避免梯度消失问题；
  - ✓ 计算简单，速度快；
  - ✓ 活跃度的分散性使得神经网络的整体计算成本下降。
- 缺点
  - ✓ 无界。
  - ✓ 梯度很大的时候可能导致的神经元“死亡”，因为很大的梯度导致更新之后的网络传递过来的输入是小于零的，从而导致 $ReLU$ 的输出是0，对应的神经元恒不更新。

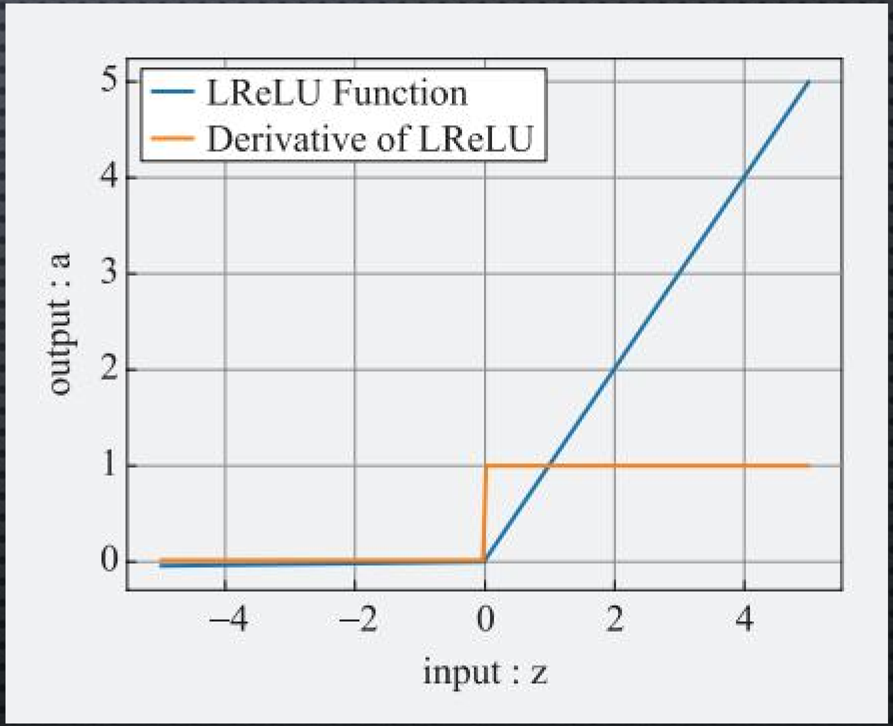
8.3 半线性激活函数

➢ ***Leaky ReLU* 函数**

- 函数表达式和导数公式

$$LReLU(z)=\begin{cases} z, z\geq 0 \\ \alpha z, z<0 \end{cases}, \quad LReLU'(z)=\begin{cases} 1, z\geq 0 \\ 0, z<0 \end{cases}$$

- 定义域和值域
  - ✓ 输入定义域：$(-\infty,\infty)$；
  - ✓ 函数输出域：$(-\infty,\infty)$；
  - ✓ 导函数输出域：$\{\alpha,1\}$。
- 优点
  - ✓ 继承了 $ReLU$ 函数的优点。
  - ✓ 在一定程度上避免神经元“死亡”。



8.3 半线性激活函数

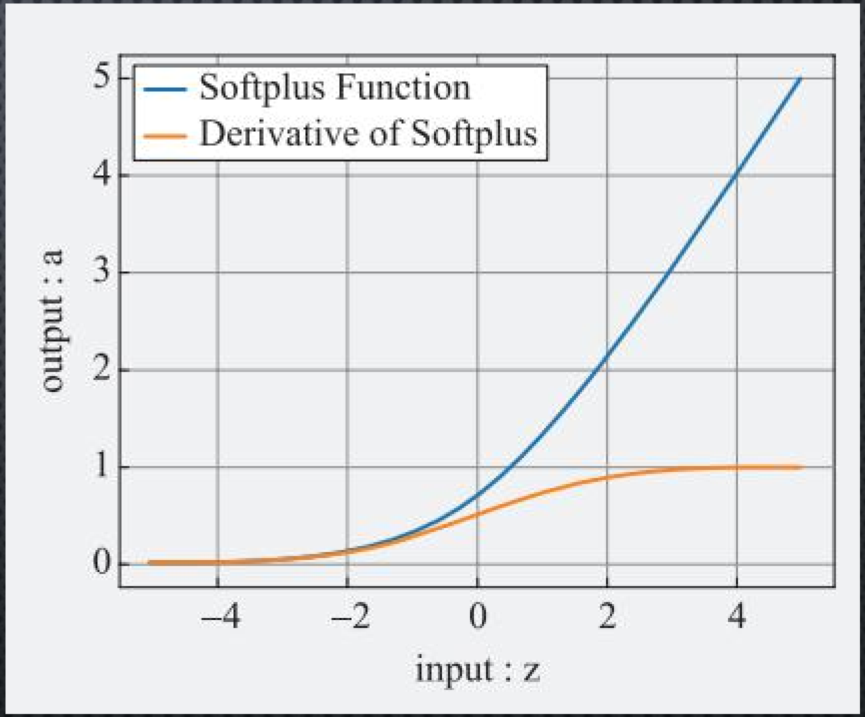
➢ *Softplus* 函数

- 函数表达式和导数公式

$$Softplus(z) = \ln(1 + e^z)$$

$$Softplus'(z) = \frac{e^z}{1 + e^z}$$

- 定义域和值域
  - ✓ 输入定义域：$(-\infty, \infty)$；
  - ✓ 函数输出域：$(0, \infty)$；
  - ✓ 导函数输出域：$(0,1)$。



8.3 半线性激活函数

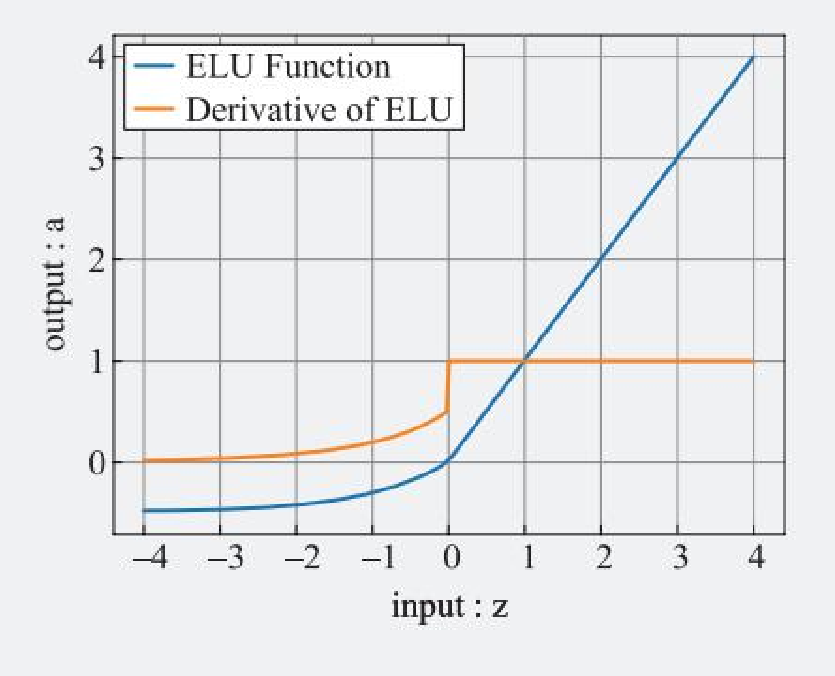
➢ *ELU* 函数

- 函数表达式和导数公式

$$ELU(z) = \begin{cases} z, z \geq 0 \\ \alpha(e^z - 1), z < 0 \end{cases}$$

$$ELU'(z) = \begin{cases} 1, z \geq 0 \\ \alpha e^z, z < 0 \end{cases}$$

- 定义域和值域
  - ✓ 输入定义域：$(-\infty, \infty)$；
  - ✓ 函数输出域：$(-\alpha, \infty)$；
  - ✓ 导函数输出域：$(0,1]$。



THE END

谢谢！