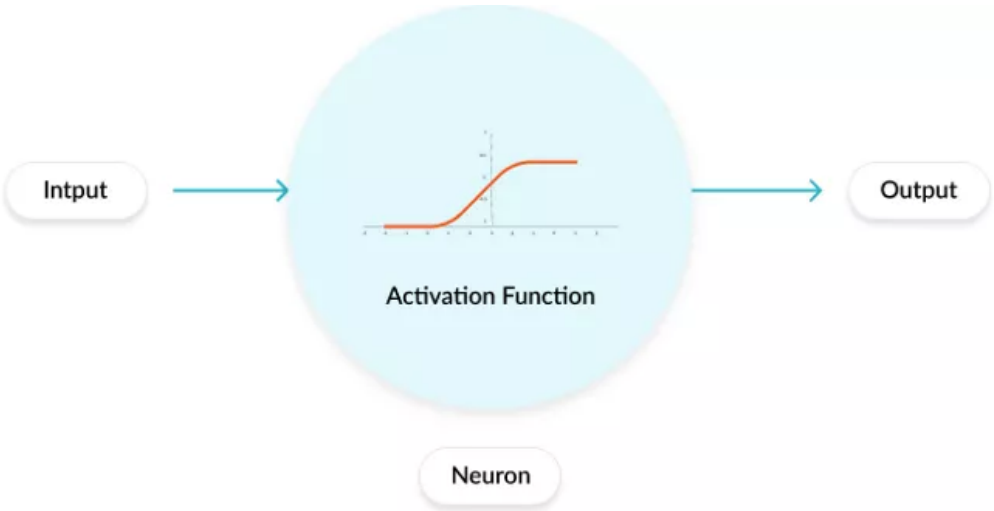
深度学习最常用的10个激活函数！（数学原理+优缺点）

小小挖掘机 2月28日

干货

作者：Sukanya Bag，来源：机器之心

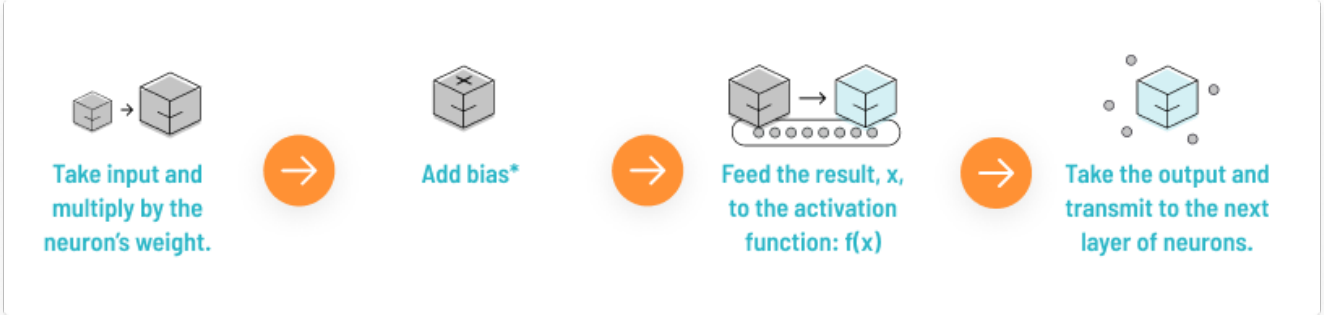
> 激活函数是神经网络模型重要的组成部分，本文作者Sukanya Bag从激活函数的数学原理出发，详解了十种激活函数的优缺点。



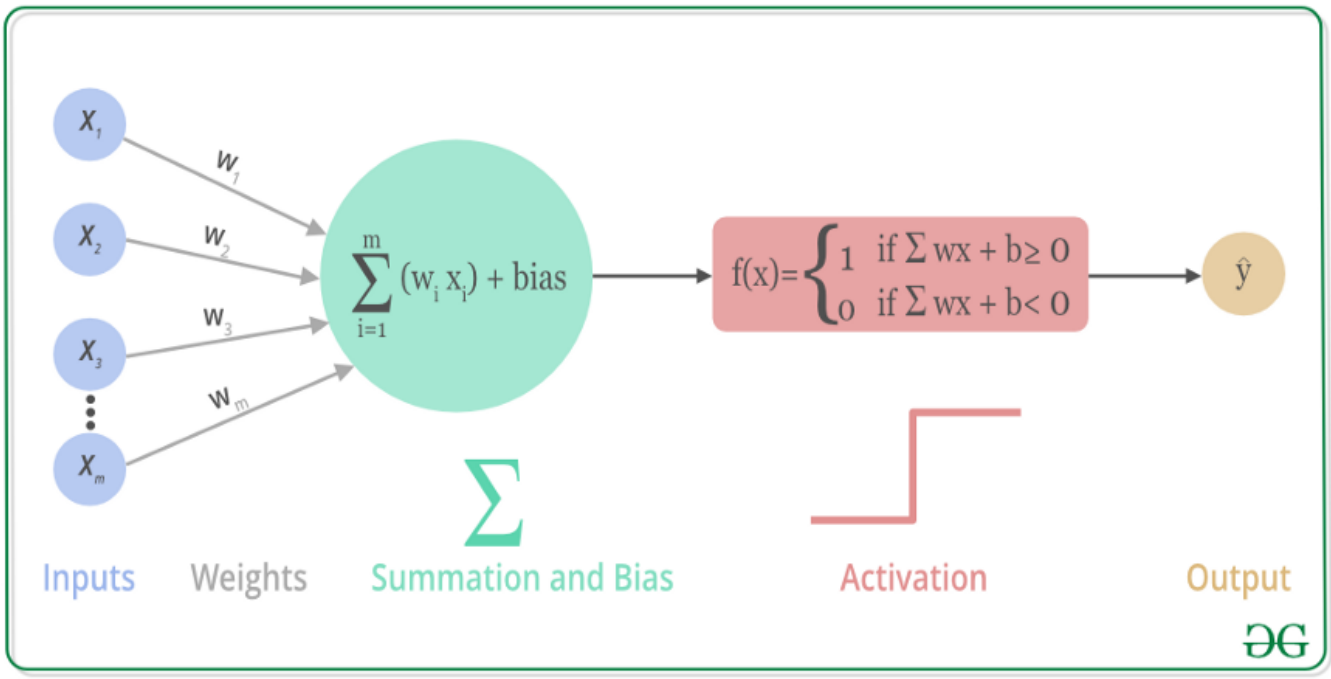
激活函数（Activation Function）是一种添加到人工神经网络中的函数，旨在帮助网络学习数据中的复杂模式。类似于人类大脑中基于神经元的模型，激活函数最终决定了要发射给下一个神经元的内容。

在人工神经网络中，一个节点的激活函数定义了该节点在给定的输入或输入集合下的输出。标准的计算机芯片电路可以看作是根据输入得到开（1）或关（0）输出的数字电路激活函数。因此，激活函数是确定神经网络输出的数学方程式，本文概述了深度学习中常见的十种激活函数及其优缺点。

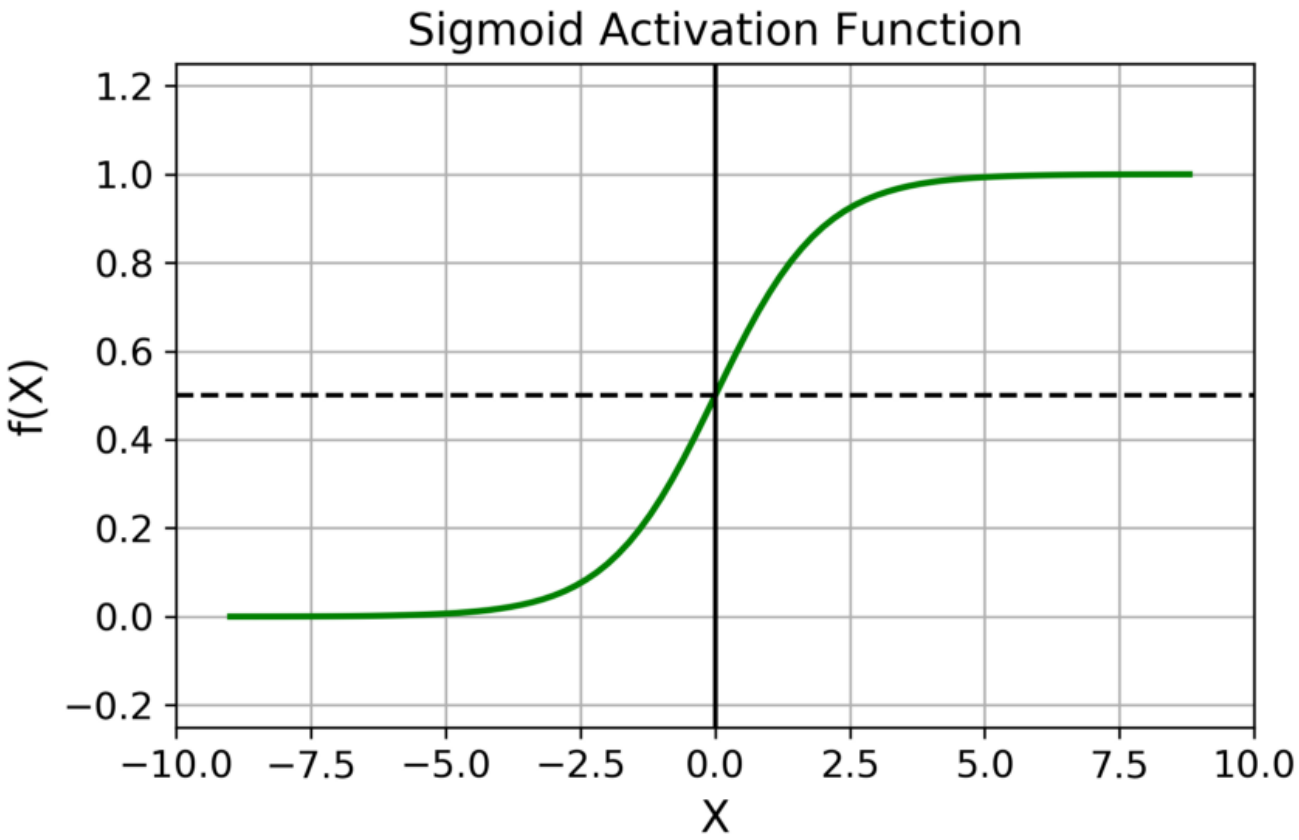
首先我们来了解一下人工神经元的工作原理，大致如下：



上述过程的数学可视化过程如下图所示：



1. Sigmoid 激活函数



Sigmoid 函数的图像看起来像一个 S 形曲线。

函数表达式如下：

$$f(z) = 1/(1+ e^{-z})$$

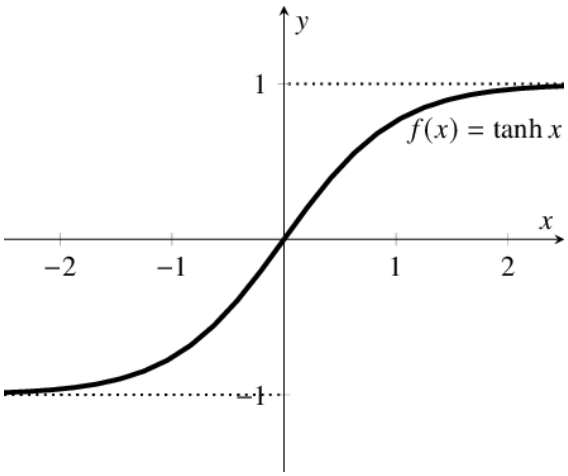
在什么情况下适合使用 Sigmoid 激活函数呢？

- Sigmoid 函数的输出范围是 0 到 1。由于输出值限定在 0 到 1，因此它对每个神经元的输出进行了归一化；
- 用于将预测概率作为输出的模型。由于概率的取值范围是 0 到 1，因此 Sigmoid 函数非常合适；
- 梯度平滑，避免「跳跃」的输出值；
- 函数是可微的。这意味着可以找到任意两个点的 sigmoid 曲线的斜率；
- 明确的预测，即非常接近 1 或 0。

Sigmoid 激活函数有哪些缺点？

- 倾向于梯度消失；
- 函数输出不是以 0 为中心的，这会降低权重更新的效率；
- Sigmoid 函数执行指数运算，计算机运行得较慢。

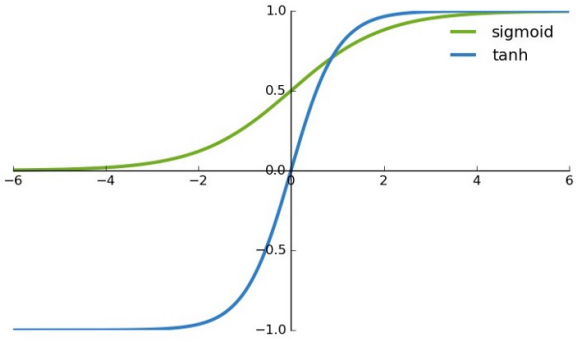
2. Tanh / 双曲正切激活函数



tanh 激活函数的图像也是 S 形，表达式如下：

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

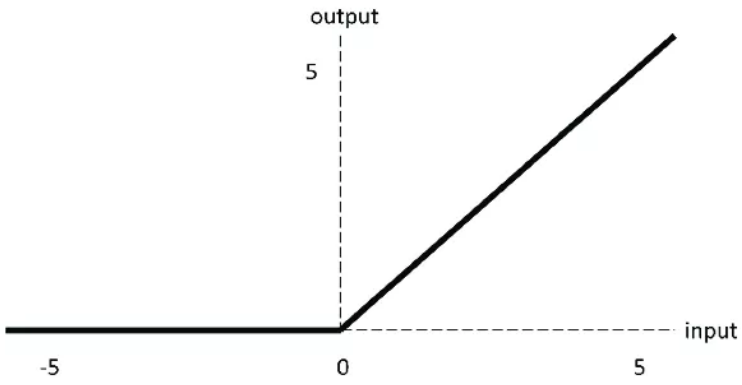
tanh 是一个双曲正切函数。tanh 函数和 sigmoid 函数的曲线相对相似。但是它比 sigmoid 函数更有一些优势。



- 首先，当输入较大或较小时，输出几乎是平滑的并且梯度较小，这不利于权重更新。二者的区别在于输出间隔，tanh 的输出间隔为 1，并且整个函数以 0 为中心，比 sigmoid 函数更好；
- 在 tanh 图中，负输入将被强映射为负，而零输入被映射为接近零。

注意：在一般的二元分类问题中，tanh 函数用于隐藏层，而 sigmoid 函数用于输出层，但这并不是固定的，需要根据特定问题进行调整。

3. ReLU 激活函数



ReLU 激活函数图像如上图所示，函数表达式如下：

$$\sigma(x) = \begin{cases} max(0, x) & , x >= 0 \\ 0 & , x < 0 \end{cases}$$

ReLU 函数是深度学习中较为流行的一种激活函数，相比于 sigmoid 函数和 tanh 函数，它具有如下优点：

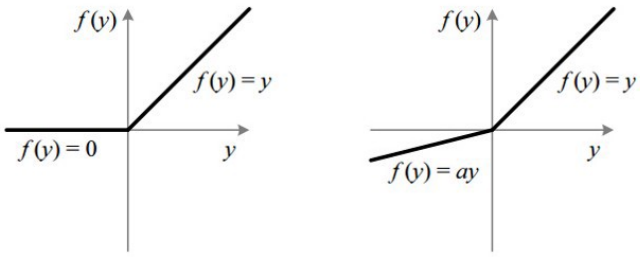
- 当输入为正时，不存在梯度饱和问题。
- 计算速度快得多。ReLU 函数中只存在线性关系，因此它的计算速度比 sigmoid 和 tanh 更快。

当然，它也有缺点：

1. Dead ReLU 问题。当输入为负时，ReLU 完全失效，在正向传播过程中，这不是问题。有些区域很敏感，有些则不敏感。但是在反向传播过程中，如果输入负数，则梯度将完全为零，sigmoid 函数和 tanh 函数也具有相同的问题；
2. 我们发现 ReLU 函数的输出为 0 或正数，这意味着 ReLU 函数不是以 0 为中心的函数。

4. Leaky ReLU

它是一种专门设计用于解决 Dead ReLU 问题的激活函数：



ReLU vs Leaky ReLU

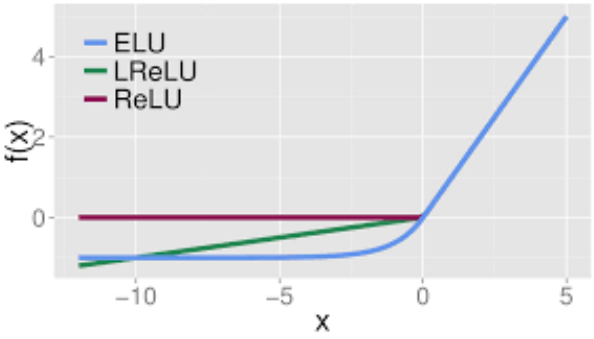
为什么 Leaky ReLU 比 ReLU 更好?

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}.$$

1. Leaky ReLU 通过把 x 的非常小的线性分量给予负输入（0.01x）来调整负值的零梯度（zero gradients）问题；
2. leak 有助于扩大 ReLU 函数的范围，通常 a 的值为 0.01 左右；
3. Leaky ReLU 的函数范围是（负无穷到正无穷）。

注意：从理论上讲，Leaky ReLU 具有 ReLU 的所有优点，而且 Dead ReLU 不会有任何问题，但在实际操作中，尚未完全证明 Leaky ReLU 总是比 ReLU 更好。

5. ELU



ELU vs Leaky ReLU vs ReLU

ELU 的提出也解决了 ReLU 的问题。与 ReLU 相比，ELU 有负值，这会使激活的平均值接近零。均值激活接近于零可以使学习更快，因为它们使梯度更接近自然梯度。

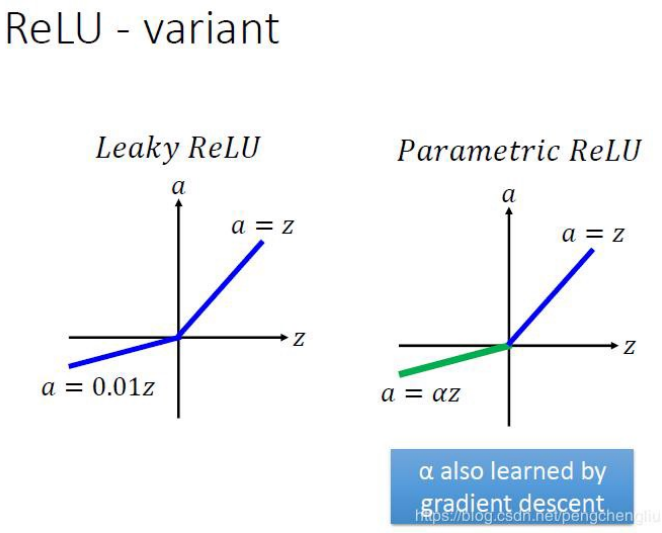
$$g(x) = \mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leqslant 0 \end{cases}$$

显然，ELU 具有 ReLU 的所有优点，并且：

- 没有 Dead ReLU 问题，输出的平均值接近 0，以 0 为中心；
- ELU 通过减少偏置偏移的影响，使正常梯度更接近于单位自然梯度，从而使均值向零加速学习；
- ELU 在较小的输入下会饱和至负值，从而减少前向传播的变异和信息。

一个小问题是它的计算强度更高。与 Leaky ReLU 类似，尽管理论上比 ReLU 要好，但目前在实践中没有充分的证据表明 ELU 总是比 ReLU 好。

6. PReLU（Parametric ReLU）



PReLU 也是 ReLU 的改进版本：

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

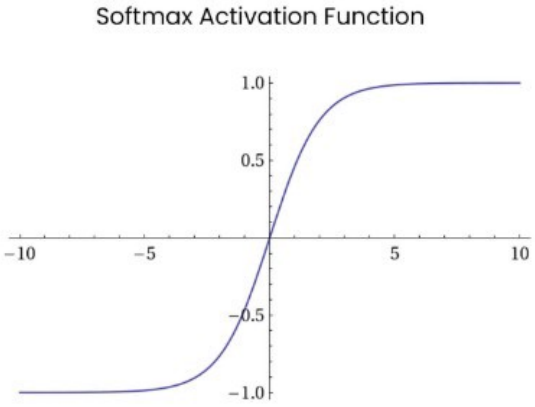
看一下 PReLU 的公式：参数α通常为 0 到 1 之间的数字，并且通常相对较小。

- 如果 a_i= 0，则 f 变为 ReLU
- 如果 a_i> 0，则 f 变为 leaky ReLU
- 如果 a_i 是可学习的参数，则 f 变为 PReLU

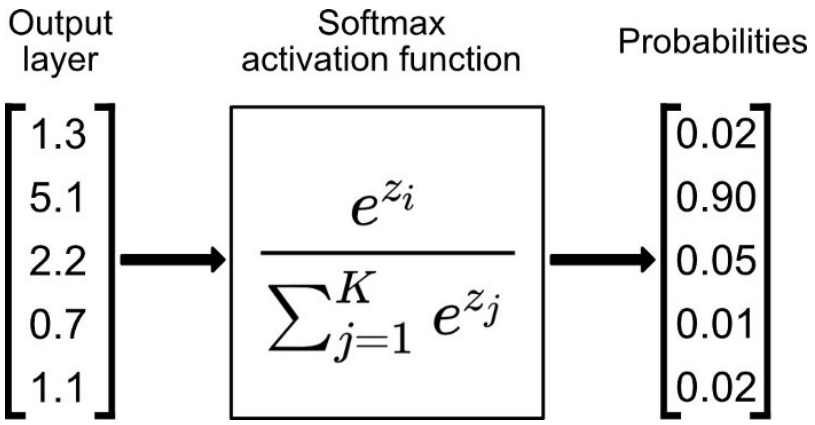
PReLU 的优点如下：

1. 在负值域，PReLU 的斜率较小，这也可以避免 Dead ReLU 问题。
2. 与 ELU 相比，PReLU 在负值域是线性运算。尽管斜率很小，但不会趋于 0。

7. Softmax



Softmax 是用于多类分类问题的激活函数，在多类分类问题中，超过两个类标签则需要类成员关系。对于长度为 K 的任意实向量，Softmax 可以将其压缩为长度为 K，值在（0，1）范围内，并且向量中元素的总和为 1 的实向量。



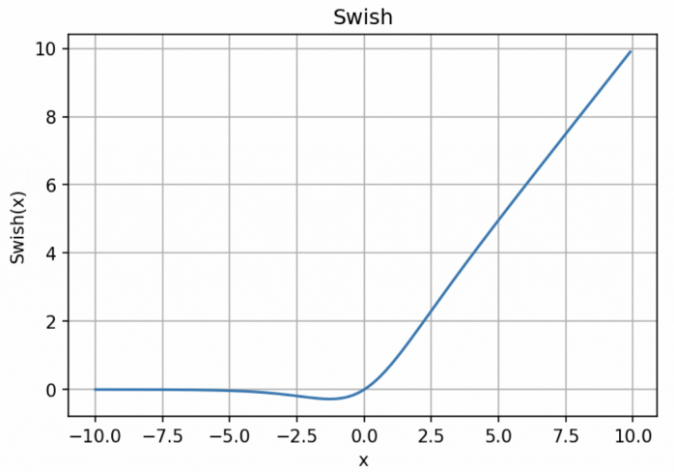
Softmax 与正常的 max 函数不同：max 函数仅输出最大值，但 Softmax 确保较小的值具有较小的概率，并且不会直接丢弃。我们可以认为它是 argmax 函数的概率版本或「soft」版本。

Softmax 函数的分母结合了原始输出值的所有因子，这意味着 Softmax 函数获得的各种概率彼此相关。

Softmax 激活函数的主要缺点是：

1. 在零点不可微；
2. 负输入的梯度为零，这意味着对于该区域的激活，权重不会在反向传播期间更新，因此会产生永不激活的死亡神经元。

8. Swish



函数表达式：y = x * sigmoid (x)

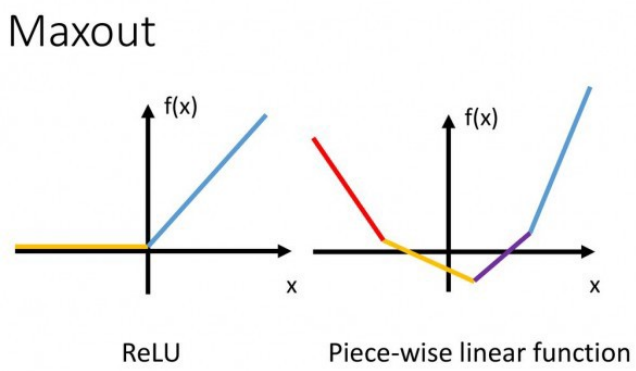
Swish 的设计受到了 LSTM 和高速网络中 gating 的 sigmoid 函数使用的启发。我们使用相同的 gating 值来简化 gating 机制，这称为 self-gating。

self-gating 的优点在于它只需要简单的标量输入，而普通的 gating 则需要多个标量输入。这使得诸如 Swish 之类的 self-gated 激活函数能够轻松替换以单个标量为输入的激活函数（例如 ReLU），而无需更改隐藏容量或参数数量。

Swish 激活函数的主要优点如下：

- 「无界性」有助于防止慢速训练期间，梯度逐渐接近 0 并导致饱和；（同时，有界性也是有优势的，因为有界激活函数可以具有很强的正则化，并且较大的负输入问题也能解决）；
- 导数恒 > 0；
- 平滑度在优化和泛化中起了重要作用。

9. Maxout

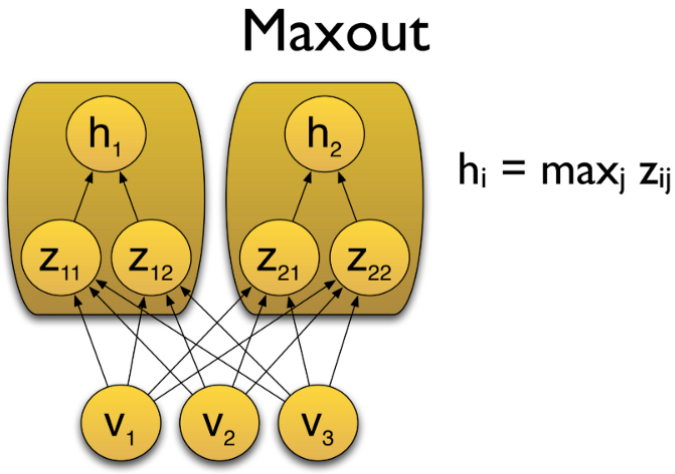


在 Maxout 层，激活函数是输入的最大值，因此只有 2 个 maxout 节点的多层感知机就可以拟合任意的凸函数。

单个 Maxout 节点可以解释为对一个实值函数进行分段线性近似（PWL），其中函数图上任意两点之间的线段位于图（凸函数）的上方。

$$ReLU = \max(0, x),\ \mathrm{abs}(x) = \max(x,\ -x)$$

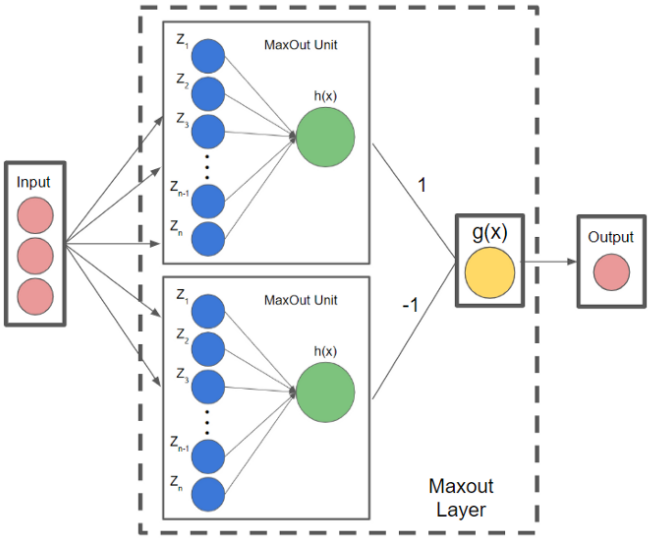
Maxout 也可以对 d 维向量（V）实现：



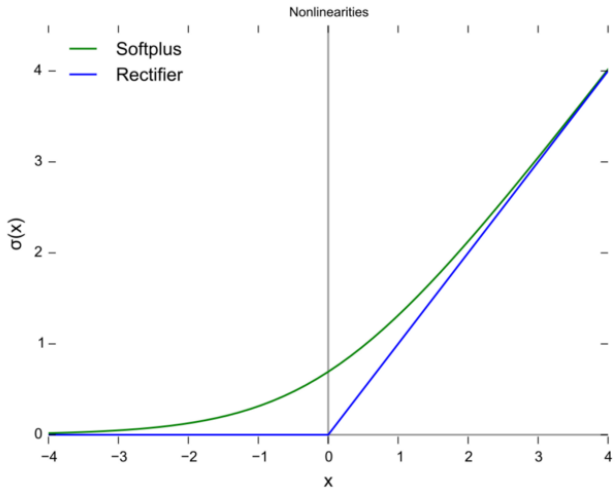
假设两个凸函数 h_1(x) 和 h_2(x)，由两个 Maxout 节点近似化，函数 g(x) 是连续的 PWL 函数。

$$g(x) = h_1(x) - h_2(x)$$

因此，由两个 Maxout 节点组成的 Maxout 层可以很好地近似任何连续函数。



10. Softplus



Softplus 函数：f（x）= ln（1 + exp x）

Softplus 的导数为

f ′(x)=exp(x) / (1+exp x)

= 1/ (1 +exp(−x))

，也称为 logistic / sigmoid 函数。

Softplus 函数类似于 ReLU 函数，但是相对较平滑，像 ReLU 一样是单侧抑制。它的接受范围很广：(0, + inf)。

原文链接：https://sukanyabag.medium.com/activation-functions-all-you-need-to-know-355a850d025e

“整理不易，**点赞三连↓**

阅读原文