激活函数

概念定义

激活函数是神经网络中用于引入非线性因素的函数。它决定了神经元是否应该被激活，即对输入信息进行“过滤”或“转换”。没有激活函数，神经网络将只能表达线性映射，其表达能力将大大受限。通过使用激活函数，可以使得深层神经网络能够学习和表示更加复杂的函数。

常见激活函数及其特点

Sigmoid函数

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

- **优点**：输出值 (0, 1) 之间，，适合用于二分类问题的概率输出。
- **缺点**：(1) 当输入值很大或很小时，梯度接近于0，导致梯度消失问题，影响模型训练速度和效果。(2) 不是0中心的，这会导致网络中的梯度更新偏向某个方向，影响训练效率。(3) 涉及指数运算，计算成本较高，尤其是在大规模数据集上。
- **应用场景**：二分类问题的输出层(如逻辑回归)。

Softmax 函数

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

- **优点**：特别适用于多分类问题，可以将多个神经元的输出转化为概率分布。
- **缺点**：计算成本较高，尤其是在处理高维数据时。
- **应用场景**：多分类问题的输出层。

Tanh (双曲正切)函数

$$\text{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- **优点**：输出值的范围是[-1, 1]，相比于Sigmoid，其均值接近于零，有助于解决梯度消失的

问题 。

- **缺点** ：和Sigmoid类似 ’当输入值很大或很小时 ’梯度接近于0 ’也会遇到梯度消失的问题 。
- **应用场景** ：隐藏层(尤其是需要对称输出时) 。

ReLU (Rectified Linear Unit) 函数

$$f(x) = max(0, x)$$

- **优点** ：(1) 计算简单快速 ’能加速网络训练过程 。(2) 解决了梯度消失的问题 ’因为对于正值 ’它的导数为1 。
- **缺点** ：存在“死亡ReLu”问题 ’对于负值 ’它的导数为0 ’这可能导致某些神经元永远不会被激活 ’即在训练过程中这部分网络无法学习 。
- **应用场景** ：隐藏层 ’尤其是深层网络 ’大多数隐藏层的默认选择 。

Leaky ReLU 函数

$$f(x) = max(\alpha x, x)$$

其中 ’α 是一个取值很小的常数 。

- **优点** ：解决了ReLU的“死亡神经元”问题 ’因为它允许负输入有一个很小的梯度 。
- **缺点** ：相比ReLU增加了计算复杂度 ’并且需要选择合适的 α 值 。
- **应用场景** ：隐藏层 ’尤其是在ReLU表现不佳时 ’可尝试Leaky ReLU 。

ELU (Exponential Linear Unit) 函数

$$f(x) = x, \text{ if } x > 0$$

$$f(x) = \alpha(e^x - 1), \text{ if } x \leq 0$$

- **优点** ：(1) 解决了ReLU的“死亡神经元”问题 ’并且相比于Leaky ReLU ’它允许负输入有一个更大的梯度 。(2) 收敛速度比ReLU更快 。
- **缺点** ：计算成本较高(涉及指数运算) 。
- **应用场景** ：隐藏层 ’需要快速收敛且计算资源充足时 。

SiLU (Sigmoid Linear Unit) 激活函数

$$\text{SiLU}(x) = x \cdot \frac{1}{1 + e^{-x}}$$

可以看出 ’SiLU对输入进行加权缩放 ’权重由 Sigmoid 函数决定 °这种设计使得 SiLU 具有平滑性和非线性特性 °

- **优点** ：(1) SiLU 的导数在大部分区域都不为 0 ’尤其是在负值区域也有梯度存在 ’因此不会出现像 ReLU 那样的“死亡”问题 °(2) 相比 Sigmoid 和 Tanh ’SiLU 更不容易出现梯度消失问题 ’因为它没有将输出严格限制在一个小范围内 °(3) SiLU 是平滑的 ’这意味着它的梯度变化较为连续 ’适合基于梯度的优化算法(如 SGD `Adam 等) °相比之下 ’ReLU 在 $x = 0$ 处不可导 ’可能导致优化过程中的不稳定 °(4) SiLU 的非线性和平滑性使其能够更好地拟合复杂的数据分布 ’从而提高模型的表达能力 °(5) 在深层网络中 ’SiLU 能够更有效地传播梯度 ’减少梯度消失或爆炸的风险 ’从而提升训练效果 °
- **缺点** ：(1) SiLU 的计算涉及指数运算 ’相比 ReLU 等简单激活函数 ’计算复杂度更高 ’增加训练时间 °(2) 由于 SiLU 的计算需要浮点运算 ’可能对某些低精度硬件(如嵌入式设备)不太友好 °(3) 虽然 SiLU 本身没有超参数 ’但它对初始化方式和优化器的选择可能更敏感 ’需要更多的调参经验 °
- **应用场景** ：隐藏层 ’高性能需求的任务(如深度学习竞赛) °