人工智能与深度学习

2-3 激活函数

王中雷

经济学院、王亚南经济研究院、邹至庄经济研究院

厦门大学

（第一版）

内容摘要

1. Sigmoid激活函数

2. Tanh激活函数

3. ReLU激活函数

4. Leaky ReLU激活函数

为什么需要激活函数?

1. 神经网络模型中，每个神经元涉及两个运算
   - 线性变换
   - 激活（非线性变换）
2. 如果没有激活（非线性变换），神经网络就是一个简单的线性模型
3. 激活运算使神经网络能够学习复杂的函数关系
4. 通用逼近定理 (Chen and Chen, 1995)：在一定条件下，神经网络模型可以以任意精度逼近任意连续函数

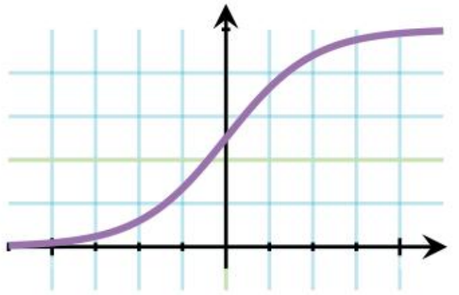
Sigmoid激活函数

1. 形式

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- 值域：$(0, 1)$
- 关于点 $(0, 1/2)$ 对称（课下自行验证）
- 导数：$\sigma'(z) = \sigma(z)\{1 - \sigma(z)\} \leq 0.25$

Sigmoid激活函数



1. 优点
   - 值域为$(0,1)$，通常用于二分类模型的输出层以估计概率
   - 当 $|z|$ 比较大时，梯度进入饱和区域，输出对输入变化不敏感
   - 对于后向传播而言，其导数计算简洁
2. 缺点
   - 梯度消失，进而导致参数更新较慢
     - 对于一个 5 层神经网络模型，梯度连乘的绝对值上界为 $0.25^5 \approx 0.001$，容易出现梯度消失
   - 指数的计算，导致该激活函数计算效率低下
   - 饱和神经元，导致模型参数更新效率低下
   - 输出是非 0 中心的，权重更新效率低下

Sigmoid激活函数

1. 为了更好地理解激活函数非零中心的缺点，考虑只有一个训练样本 $(\boldsymbol{x}, y)$ 的情况
2. 在第 2-2 节中，我们已经展示了 $\mathrm{d}\boldsymbol{W}^{[2]} = \mathrm{d}z^{[2]} \cdot (\boldsymbol{a}^{[1]})^{\mathrm{T}}$
   - $\boldsymbol{a}^{[1]} \in \mathbb{R}^{4\times 1}$
   - $\mathrm{d}z^{[2]} = a^{[2]} - y$
3. 如果第一隐藏层用的激活函数为 sigmoid 函数，那么，$\boldsymbol{a}^{[1]}$ 中的每个元素都是正数
4. 那么，$\mathrm{d}\boldsymbol{W}^{[2]}$ 中每个元素的符号，将仅仅由一个标量 $\mathrm{d}z^{[2]}$ 决定
5. 换言之，$\mathrm{d}\boldsymbol{W}^{[2]}$ 所有元素的符号与 $\mathrm{d}z^{[2]}$ 相同
6. 这将导致在更新 $\boldsymbol{W}^{[2]}$ 时，出现低效的 “锯齿形下降路径”

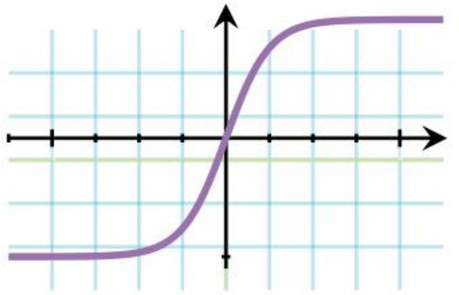
Tanh激活函数

1．形式

$$\tanh(z) = \frac{2}{1+\exp(-2z)} - 1$$

- $\tanh(z) = 2 \cdot \sigma(2z) - 1$
- 值域：$(-1, 1)$
- 关于原点 $(0, 0)$ 对称（课下自行验证）
- 导数：$\tanh'(z) = 1 - \tanh^2(z) \leq 1$

Tanh激活函数



1．优点

- 激活函数输出是 0 中心的
- 相对于 sigmoid 激活函数，tanh 激活函数在 0 附近的导数更大
- 值域是有界的：$(-1, 1)$

2．缺点

- 梯度消失
- 指数导致计算效率低下
- 饱和神经元，导致模型参数更新效率低下

ReLU激活函数

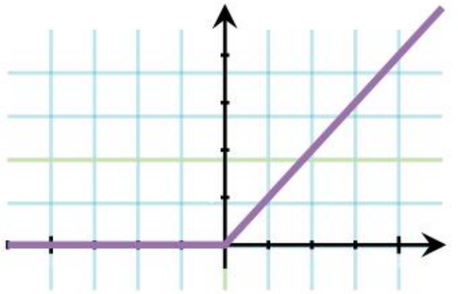
1. 形式 (Rectified Linear Unit)

$$\mathrm{ReLU}(z) = \max\{0, z\}$$

- 值域：$[0, \infty)$
- 导数：

$$\mathrm{ReLU}'(z) = \begin{cases} 0 & z \leq 0 \\ 1 & z > 0 \end{cases}$$

ReLU激活函数



1. 优点
   - 计算效率高
   - 缓解了 sigmoid、tanh 等激活函数引起的梯度消失问题
   - 稀疏激活
2. 缺点
   - 死亡 ReLU (Dying ReLU)
     - $\mathrm{d}\boldsymbol{W}^{[1]} = \mathrm{d}\boldsymbol{z}^{[1]} \cdot (\boldsymbol{x})^{\mathrm{T}}$
   - 输出是非 0 中心的
   - 在 0 处不可导，但存在次梯度
   - 在正半轴上无界

ReLU激活函数

1. 问题: 在训练模型的过程中，如果有 40% 的神经元 “死亡”
   - 可能的原因是什么?
   - 这一定是坏事吗?
   - 在必要的情况下，如何 “复活” 这部分神经元呢?
2. 答案:
   - 过大的学习率或不恰当的参数初始化
   - 不一定是坏事
   - 使用较小的学习率以及不同的初始化

Leaky ReLU激活函数

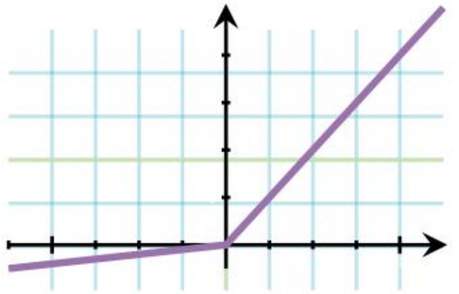
1．形式 (Leaky Rectified Linear Unit)

$$\text{LeakyReLU}(z) = \begin{cases} \alpha z & z \leq 0 \\ z & z > 0 \end{cases}$$

- α：较小的正常数。例如，$\alpha = 0.1$
- 值域：$(-\infty, \infty)$
- 导数：

$$\text{LeakyReLU}'(z) = \begin{cases} \alpha & z \leq 0 \\ 1 & z > 0 \end{cases}$$

Leaky ReLU激活函数



1．优点

- 缓解了 dying ReLU 的问题
- 计算简单
- 算法收敛速度较快
- 没有饱和区域

2．缺点

- α 的值较难确定
- 输出是非 0 中心的，权重更新效率低下
- 稀疏性略有减少

其他激活函数

1. 其他激活函数包括
   - PReLU
   - ELU
   - SELU
   - Swish (SiLU)
   - GELU
   - Mish
   - Softmax
   - $\cdots$
2. 请大家课下自行学习

激活函数选择建议

1. ReLU（或者其变体）是隐藏层的默认选择
2. 当 ReLU 出现问题时，我们可以选择 Leaky ReLU 或者其他变体
3. 当我们需要概率作为输出时，我们考虑 sigmoid 或者 softmax
4. 回归问题的输出层通常采用线性（恒等）激活函数
5. 如有其他问题，我们需进行实验分析

课程总结

1．激活函数选择的四个标准

- 非线性
- 零中心
- 计算效率
- 任务输出需求