# 深度学习

**作者：Calvin**
**QQ:** 179209347
**Mail:** 179209347@qq.com

# 介绍

**笔记简介：**

- 面向对象：深度学习初学者
- 依赖课程：**线性代数，统计概率**，优化理论，图论，离散数学，微积分，信息论

**知乎专栏：**

https://zhuanlan.zhihu.com/p/693738275

**Github & Gitee 地址：**

https://github.com/mymagicpower/AIAS/tree/main/deep_learning

https://gitee.com/mymagicpower/AIAS/tree/main/deep_learning

*** 版权声明：**

- 仅限用于个人学习
- 禁止用于任何商业用途

# 激活函数

激活函数在神经网络中扮演着至关重要的角色，它们通常袈用于神经元的输出，将神经元的输入转换为输出信号。

## 激活函数的作用

激活函数在神经网络中扮演着至关重要的角色，不仅可以增加网络的表达能力，还可以解决梯度消失问题，提高网络的学习效率和性能。激活函数引入非线性变换，使神经网络可以学习复杂的非线性关系。

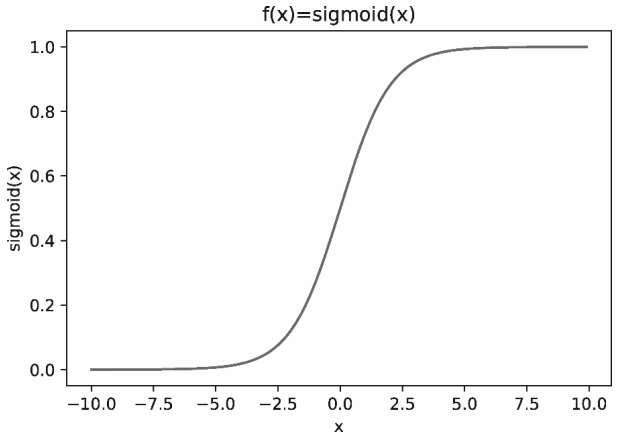
- 引入非线性性
- 解决梯度消失问题
- 增加网络的表达能力
- 稀疏性和抑制

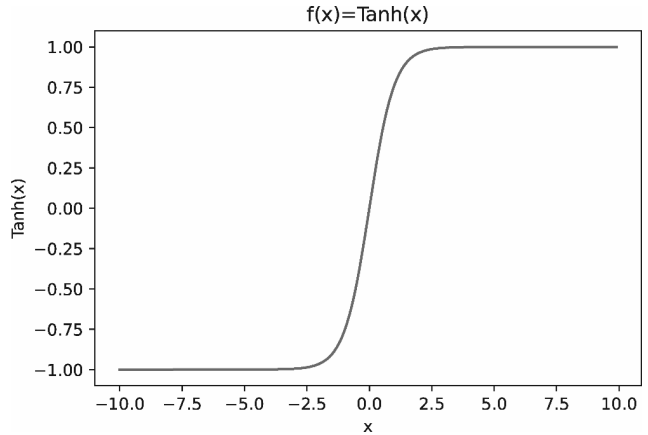
常用的激活函数有Sigmoid函数、Tanh函数、ReLU函数和Leaky ReLU函数等。

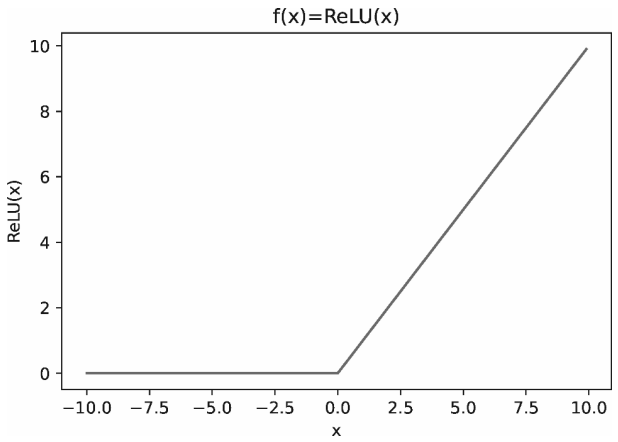
# 激活函数的性质

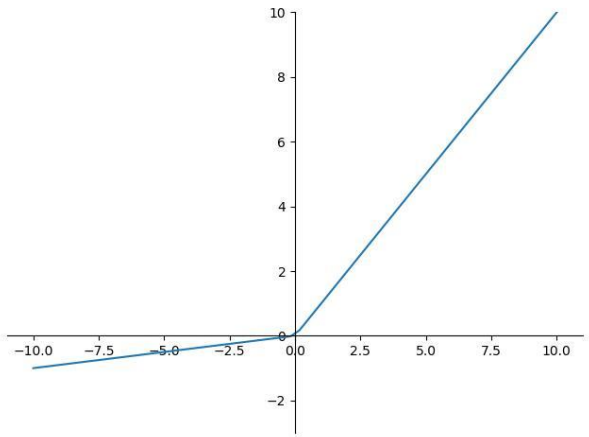
以下是激活函数的一些重要性质总结要点：

- **可微性**：梯度计算依赖于激活函数的导数。
- **非饱和性**：饱和指的是激活函数在输入较大或较小时导数趋近于0，导致梯度消失问题。
- **单调性**：单调激活函数有助于简化优化问题。
- **输出范围**：固定的输出范围，有助于控制神经元输出的幅度。
- **计算效率**：激活函数的计算效率对于深度神经网络的训练和推理速度至关重要。
- **鲁棒性**：激活函数应该对于不同范围的输入都能够表现良好。



# 常见激活函数及其导数

| 激活函数 | 函数 | 导数 |
| --- | --- | --- |
| Sigmoid函数 | $f(x)=\frac{1}{1+\exp(-x)}$ | $f'(x)=f(x)\big(1-f(x)\big)$ |
| Tanh函数 | $f(x)=\frac{1-\exp(-2x)}{1+\exp(-2x)}$ | $f'(x)=1-f\big(x\big)^2$ |
| ReLU函数 | $f(x)=\max(0,x)$ | $f'(x)=\{1,x>0;0,x<=0\}$ |
| Leaky ReLU函数 | $f(x)=\begin{cases}x & x>=0\\ ax & x<0,\ 0<\ a<1\end{cases}$ | $f'(x)=\{1,x>=0;a,x<0\}$ |

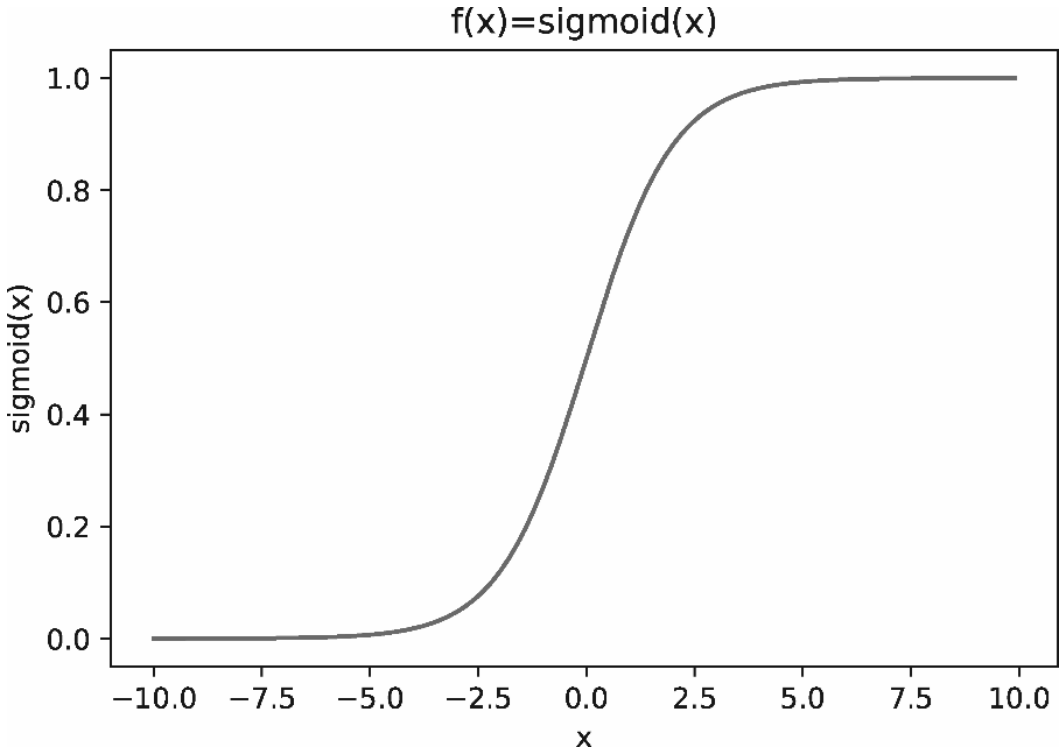
# Sigmoid函数

Sigmoid函数的定义如下:

$$f(x) = \text{Sigmoid}(x) = \frac{1}{1+\exp(-x)}$$

导数

$$f'(x) = f(x)\big(1 - f(x)\big)$$




Sigmoid函数的图形

# Sigmoid函数

## Sigmoid函数优点

- 平滑性：作为连续可导的函数，具有平滑的特性，这使得在梯度下降等优化算法中更容易进行计算。
- 易于求导：导数可以用函数本身来表示，这简化了梯度计算的过程，有利于神经网络的训练。
- 输出范围有界：输出范围在(0, 1)之间，可以将输出解释为概率值，适用于二分类问题。
- 非线性：作为一种非线性函数，可以帮助神经网络学习复杂的模式和关系。
- 相对简单：表达式简单明了，计算也相对容易，适合在一些简单的神经网络结构中使用。

## Sigmoid函数缺点

- 梯度消失问题：导数在输入接近正无穷或负无穷时会趋于零，导致梯度消失问题。
- 输出不以零为中心：输出范围在(0, 1)之间，且不以零为中心。
- 计算代价较高：Sigmoid函数的计算相对复杂，涉及指数运算，这会增加计算的复杂度。

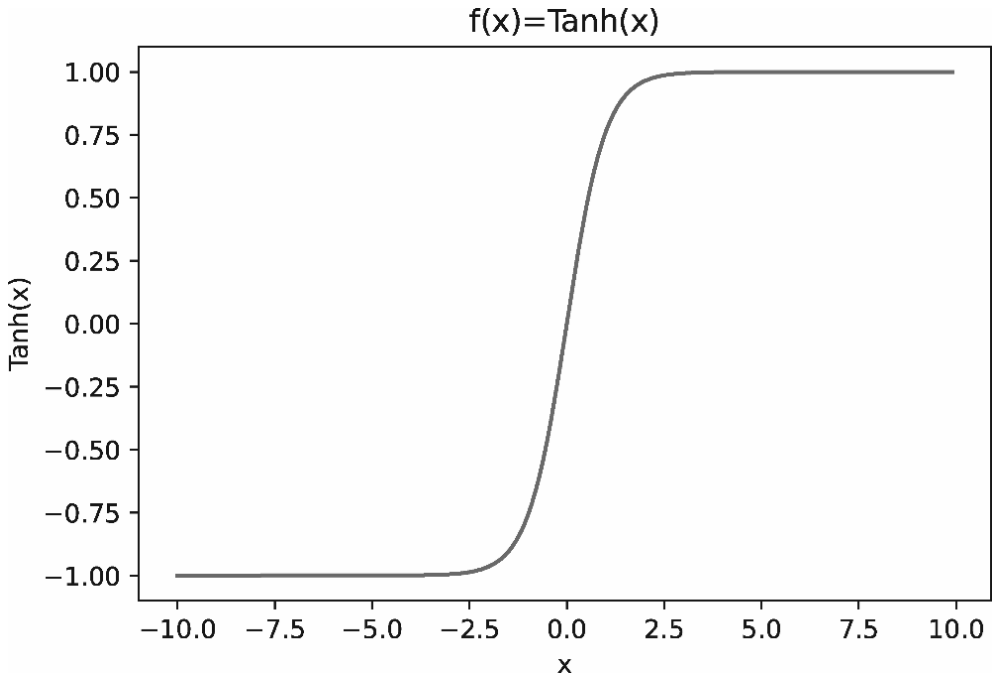
# Tanh函数

Tanh函数的定义如下:

$$f(x) = Tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

导数

$$f'(x) = 1 - f(x)^2$$




Tanh函数的图形

与Sigmoid函数一样，Tanh函数也是在神经网络中较早得到应用的激活函数。

# Tanh函数

## Tanh函数优点

- 当输入过大或过小时，输出几乎是平滑的，梯度小，不利于权值的更新。区别在于输出间隔。
- tanh的输出区间为1，整个函数以0为中心，优于sigmoid。
- 其主要优点是负数输入将被映射为接近-1，而零输入将被映射为tanh图中接近零的地方。

## Tanh函数缺点

- 梯度消失问题：在使用深度神经网络时，Tanh函数容易导致梯度消失或梯度爆炸的问题。
- 计算复杂度高：Tanh函数的计算复杂度相对较高，因为它涉及指数运算。

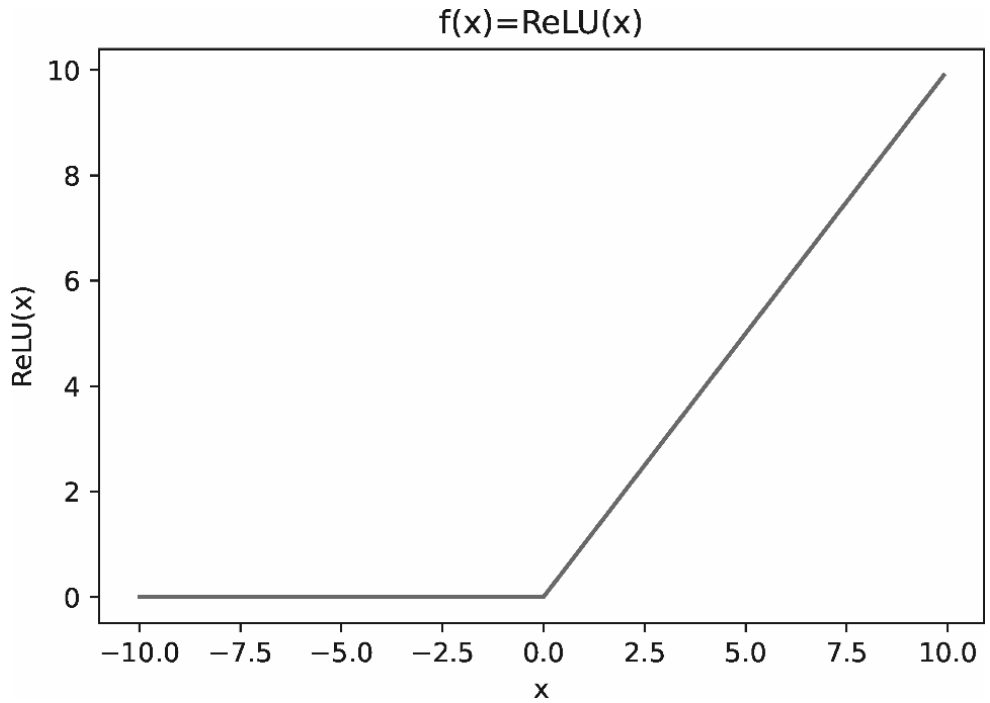
# ReLU函数

ReLU 函数的定义如下:

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

导数

$$f'(x) = \{1, x > 0; 0, x <= 0\}$$




ReLU函数的图形

ReLU函数是近几年才得到应用的激活函数。相较于Sigmoid函数和Tanh函数，ReLU函数可以提供更好的结果。

# ReLU函数

## ReLU函数优点

- 计算简单：ReLU函数的计算非常简单，只需要比较输入是否大于零。
- 稀疏激活性： 当输入为负时，ReLU函数的输出为零，这种稀疏激活性有助于模型的稀疏性，提高模型的泛化能力。
- 解决梯度爆炸问题：ReLU函数在正区间上的导数为常数1，可以避免梯度爆炸问题，有助于训练深层神经网络。
- 推动稀疏表示学习： ReLU函数的稀疏性有助于推动神经网络学习到更加有效的特征表示，提高模型的泛化能力。

## ReLU函数缺点

- 神经元死亡问题：在训练过程中，某些神经元可能永远不会被激活，导致这些神经元对应的权重永远无法更新。
- 梯度消失：在反向传播过程中，当输入值为负时，ReLU的梯度为0，这可能导致梯度消失问题，使得权重无法得到有效更新，从而影响模型的训练效果。
- 不是处处可导：ReLU函数在零点处不可导，这可能导致一些优化算法无法使用。
- 不对称性：ReLU函数是非线性的，并且在负半轴上完全不活跃。这种不对称性可能导致模型训练时出现一些问题，特别是对称性相关的任务。
- 不适用于输出层：ReLU函数的输出范围为$[0, +\infty)$，这使得它不适用于需要输出范围在特定区间内的任务，如分类问题中的多类别输出。

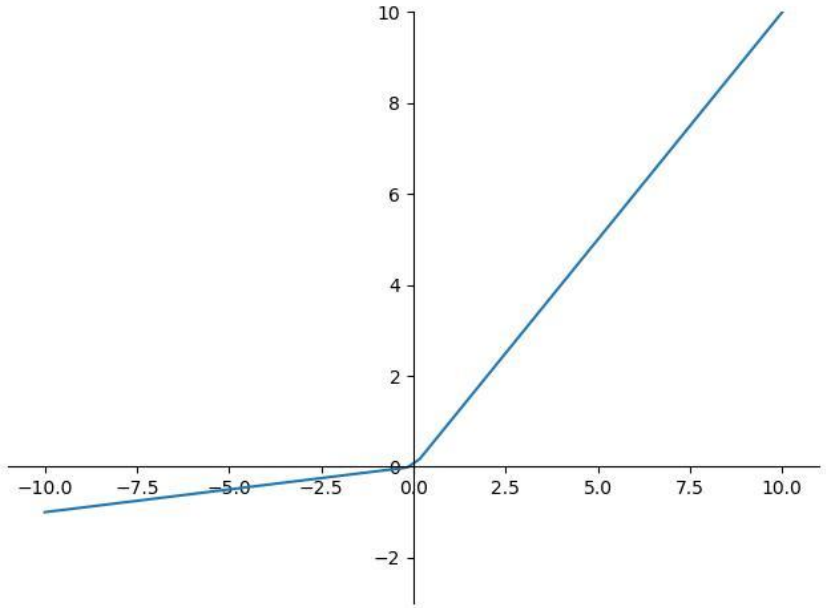
# Leaky ReLU函数

ReLU 函数的定义如下:

$$f(x) = Leaky\ RelU(x) = \begin{cases} x & x >= 0 \\ ax & x < 0, 0 < a < 1 \end{cases}$$

导数

$$f'(x) = \{1, x >= 0; a, x < 0\}$$




Leaky ReLU函数的图形

# Leaky ReLU函数

Leaky ReLU**函数优点**

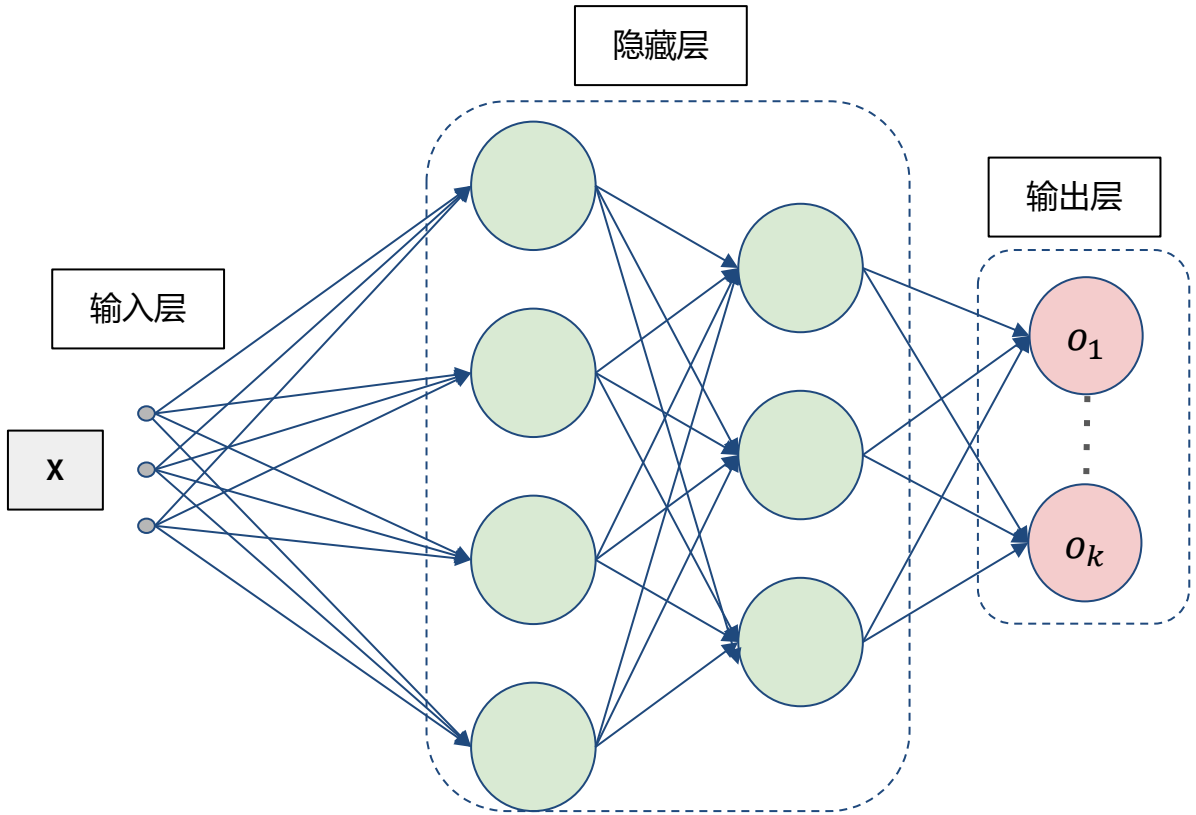
- Leaky ReLU函数具备ReLU的所有优点，并且一定程度缓解了神经元死亡问题。

Leaky ReLU**函数缺点**

- 负输入仍可能导致神经元失活：虽然Leaky ReLU允许负数输入有一个小的梯度，但是对于极端负数输入，仍然会导致神经元失活。这可能会导致梯度消失问题，使得神经元无法学习。
- 不具备单调性：Leaky ReLU并不是严格的单调递增函数，因为它的斜率在负数区域是固定的。这可能会导致一些优化问题，使得模型训练变得更加困难。
- 参数选择困难：Leaky ReLU函数中的斜率参数通常需要手动调整，这可能会增加超参数调整的复杂性。选择不当的斜率参数可能会影响模型的性能，需要耗费额外的时间和精力来进行调优。

# Softmax函数 - 多分类

深度学习在多分类任务中扮演着重要的角色，它是一种机器学习方法，通过模拟人类大脑的神经网络结构来学习数据的特征表示。在多分类问题中，我们的目标是将输入数据分为两个以上的类别。
在多分类问题的输出层通常会使用Softmax激活函数，将神经网络的输出转化为各个类别的概率分布。
Softmax函数可以确保所有类别的输出概率之和为1，便于解释和比较。




$$y_1, y_2, \ldots, y_k = \text{softmax}(o_1, o_2, \ldots o_k)$$

# Softmax函数

Softmax函数适用于多元分类问题，它的定义如下:

$$\text{Softmax}(o_i) = \frac{\exp(o_i)}{\sum_k \exp(o_j)}$$

$o_i$ 为第 $i$ 个结点的输出值， $k$ 为输出结点的个数，即分类的类别个数。通过Softmax函数可以将多分类的输出值转换为范围在 [0,1]，且和为1的概率分布。


Thank
You