

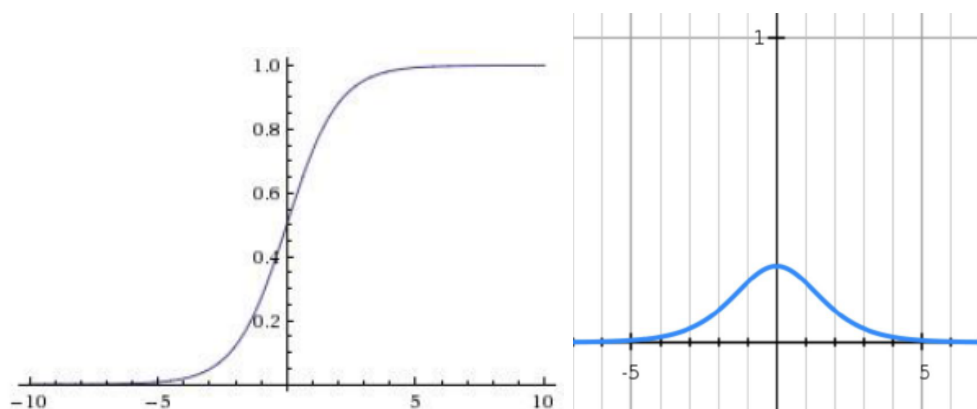
目录

1 Activation	1
1.1 Sigmoid	1
1.2 tanh	1
1.3 ReLU: Rectified Linear Unit	1
1.4 Leaky ReLU	2
1.5 ELU: Exponential Linear Units	2
1.6 SELU	2
1.7 Maxout	2

1 Activation

1.1 Sigmoid

Sigmoid 函数定义如下: $\sigma(x) = \frac{1}{1+e^{-x}}$, 则其导函数为: $\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \sigma(x) \cdot (1 - \sigma(x))$, Sigmoid 函数的图形和其导数的图形如图所示:



该函数将输入从 $[-\infty, \infty]$ 映射到 $[0, 1]$, 但是它存在着3个问题:

- 其饱和特性会抑制神经元的梯度;
- 其输出并不以0为中心, 假设神经元的输入总是为正(Sigmoid的输出本身就总是为正), 那么其权重的梯度便总是与upstream gradient的符号相同;
- 指数运算代价大;

1.2 tanh

tanh 函数定义如下: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, 其导函数为: $\tanh'(x) = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^4}$

该函数将输入从 $[-\infty, \infty]$ 映射到 $[-1, 1]$, 虽然输出以零为中心, 但是仍然有饱和的问题。

1.3 ReLU: Rectified Linear Unit

该函数不会饱和, 并且计算高效, 实际使用中收敛速度明显快于sigmoid与tanh, 但它的输出并不以0为中心。

1.4 Leaky ReLU

Leaky ReLU = $\max(0.01x, x)$ ，该函数有着ReLU的优点，且反向传播时梯度不会直接变成0。若将0.01设置成其他的参数，便是PReLU = $\max(\alpha x, x)$

1.5 ELU: Exponential Linear Units

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$$

该函数与Leaky ReLU相比，在负饱和的地方增加了对噪声的鲁棒性，但是需要指数计算，因此计算效率低。

1.6 SELU

$$f(x) = \begin{cases} \lambda x, & x > 0 \\ \lambda \alpha(e^x - 1), & x \leq 0 \end{cases}$$

在深度学习中效果更好，具有自归一化的作用。

1.7 Maxout

$\text{out} = \max(w_1^T x + b_1, w_2^T x + b_2)$ ，ReLU和Leaky ReLU是其特例，不饱和也不陷入死区，但参数增加了。

实际中，通常应该先使用ReLU，然后尝试更换成Leaky ReLU / Maxout / ELU / SELU等来获得小幅度的性能提升，应该避免使用sigmoid和tanh。