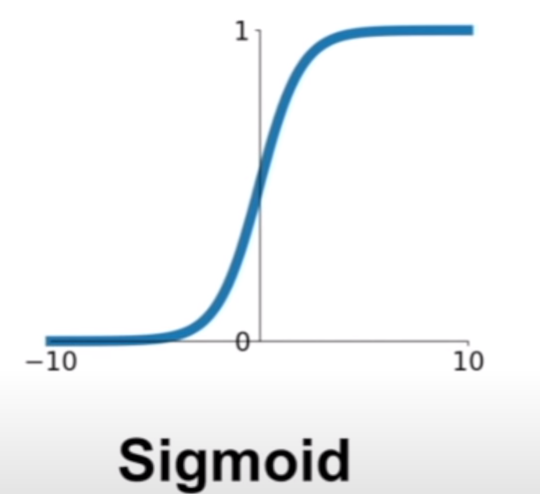
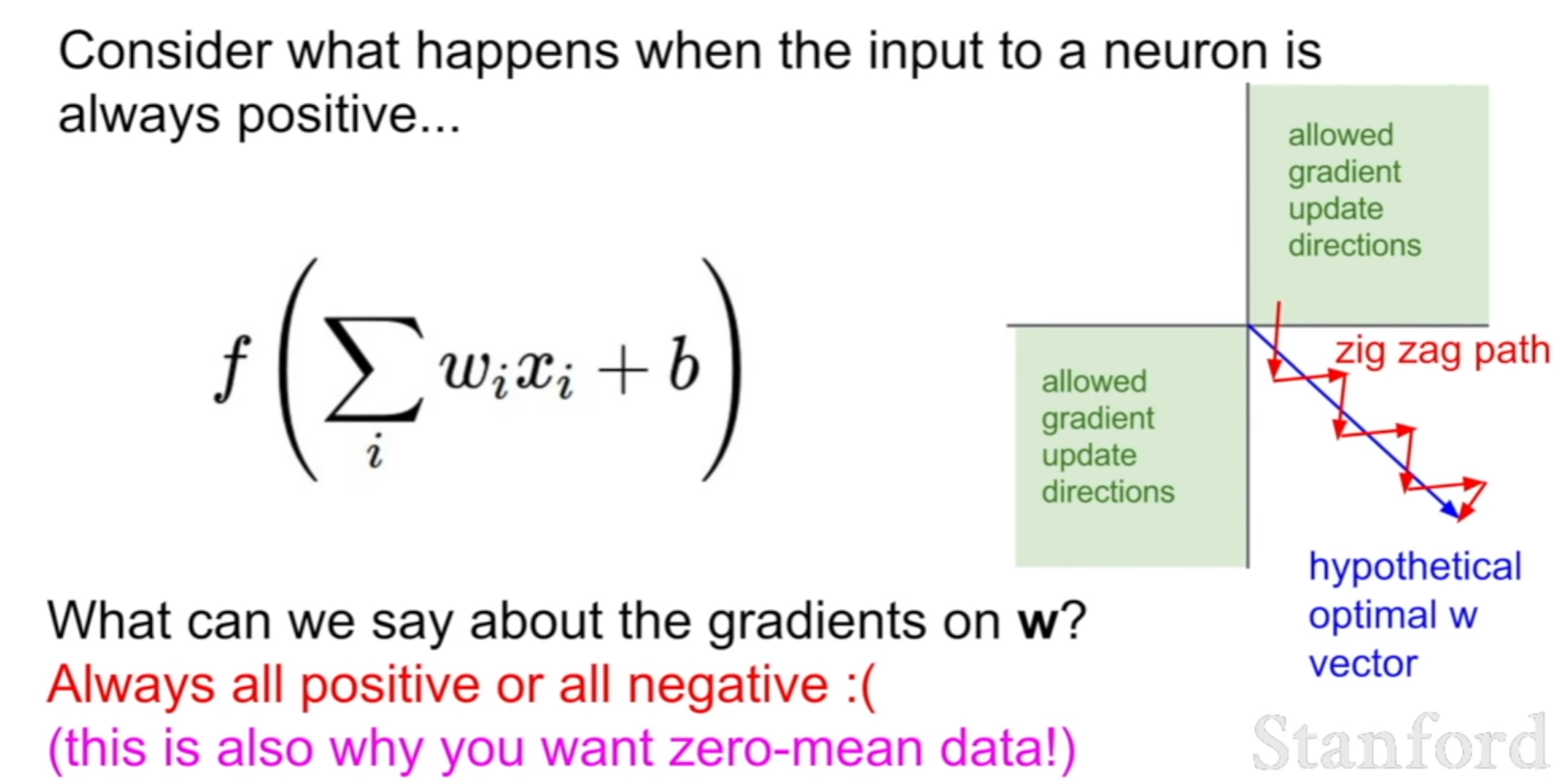
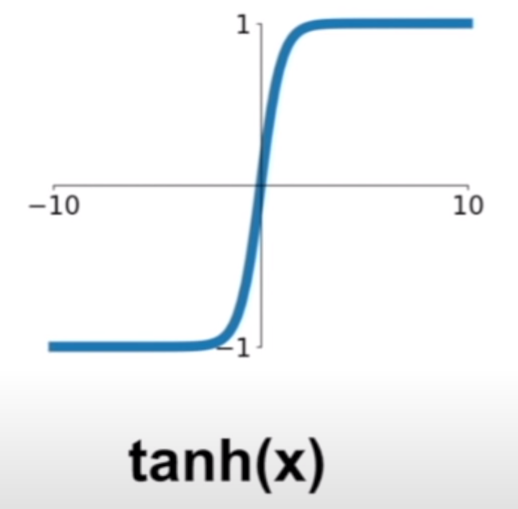


1. **Sigmoid:** 

**优点**：（1）把所有的数都压缩到了[0, 1]区间，非常大或者非常负的值都会被压缩成常数，中间部分近似线性；（2）过去很流行，因为sigmoid可被理解为一个神经元的saturating “firing rate”。

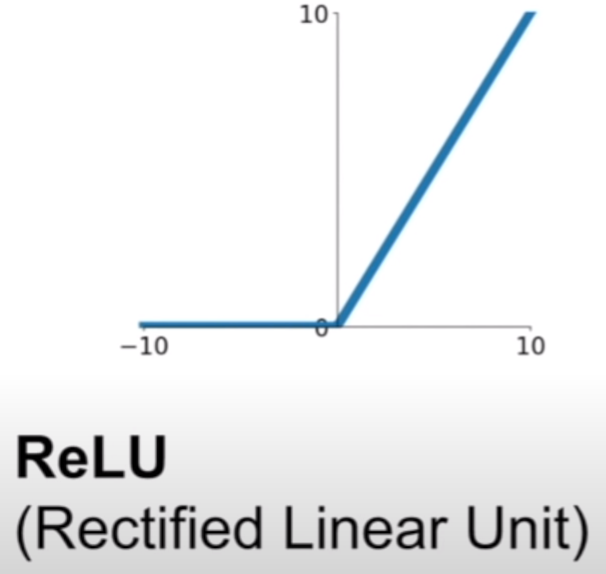
**缺点：**（1）饱和神经元会产生梯度消失，因为在两端是常数，梯度为0 。（2）没有以0为曲线的分布中心。为什么？dL/dW=dL/df\*df/dW，其中df/dW=X，而X则是来自sigmoid的输出，所以X总是正的。那么dL/dW的符号就是由upstream dL/df决定的，dL/df的值可负可正，***但是为什么这里要说W的梯度要么总是正的要么总是负的***?下面右图用来说明总是负或正的对梯度更新的影响。对于二维W，这种限制就会导致递归往下面红色箭头的方向锯齿型移动，而无法沿着最后的蓝色向量方向进行梯度下降。（3）公式中的指数计算成本高。



1. **Tanh**：

**优点**：（1）把数据压缩到[-1, 1]区间；（2）曲线分布以0为中心；（3）中间部分近似线性；

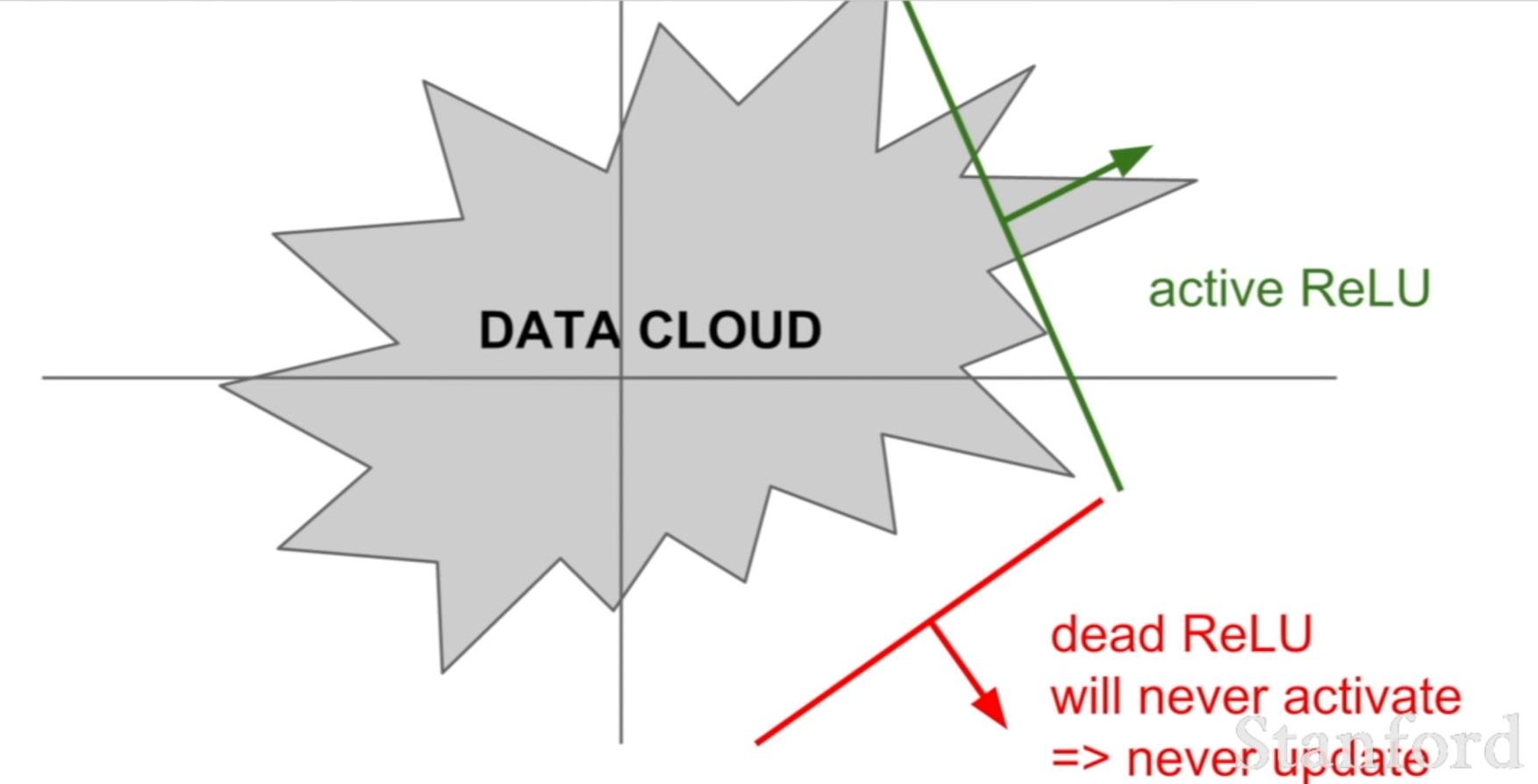
**缺点**：（2）梯度消失。

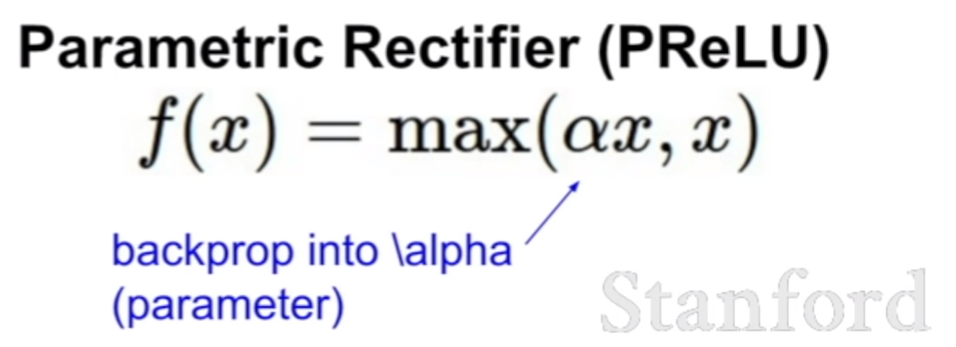
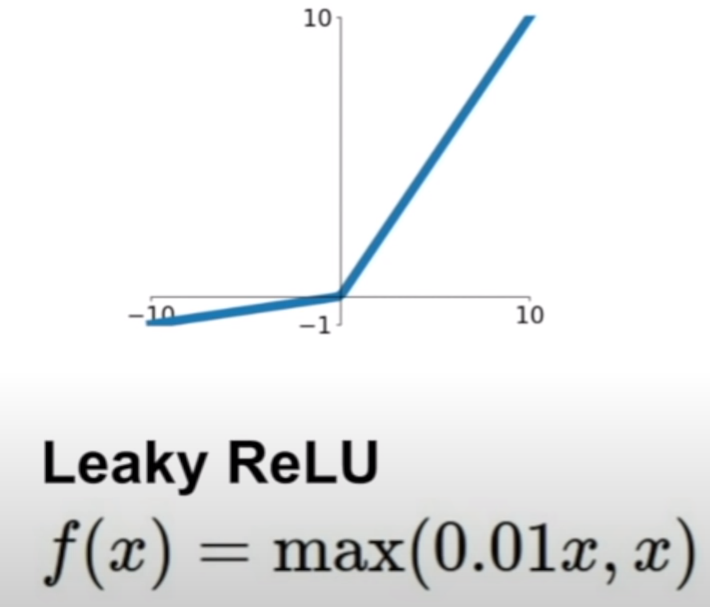
1. **ReLU：**

**优点**：（1）不会出现饱和情况；（2）计算效率高；（3）比sigmoid/tanh收敛快；

**缺点**：（1）不是以0为中的分布；（2）左半部对于x<0的数据会出现梯度消失的情况，有些神经元会die。什么时候会导致这种情况？第一种是bad weights initialization，如初始化的部分权重处于下图中的右下角区域。第二种是learning rate is too high导致的，因为如果 梯度下降跨度太大，即使在训练初期表现好，后期也可能跨过下面的边界线导致die out。

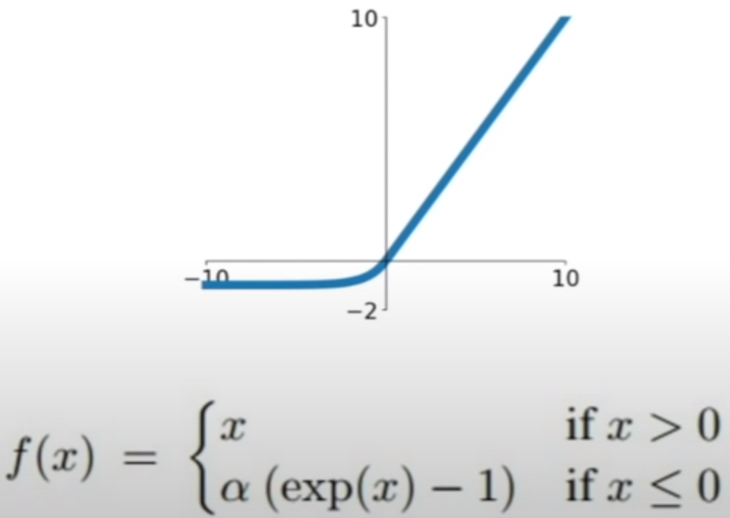
**注**：在实际情况中，人们倾向用小的positive biases，如0.01来初始化ReLU神经元。



1. **Leaky ReLU：**

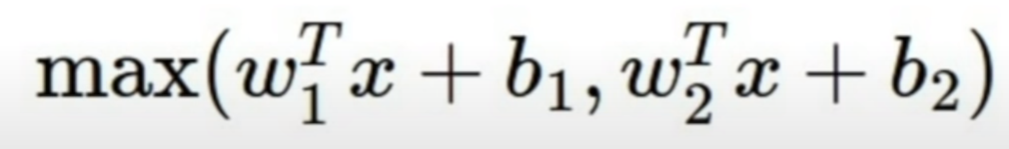
**优点**：（1）不会出现饱和情况；（2）计算效率高；（3）比sigmoid/tanh收敛快；（4）神经元不会die。

**缺点**：（1）不是以0为中的分布；（2）左半部对于x<0的数据会出现梯度消失的情况，有些神经元会

1. **ELU:** 

**优点**：（1）不会出现饱和情况；（2）计算效率高；（3）比sigmoid/tanh收敛快；（4）更接近以0为中心的曲线分布；（5）负饱和机制增加了对噪声的鲁棒性（相比Leaky ReLU）

**缺点**：（1）计算需要用到指数运算。

1. **Maxout “Neuron”:** 

**优点**：（1）泛化ReLU和Leaky ReLU；（2）线性机制，不会饱和，不会die

**缺点**：产生两倍的参数/神经元

**实际应用中应该如何选择激活函数？**

1. 优先使用ReLU，要注意learning rate的设置。
2. 尝试Leaky ReLU / Maxout / ELU
3. 可尝试tanh，但是别期望太多
4. ***不要使用sigmoid***