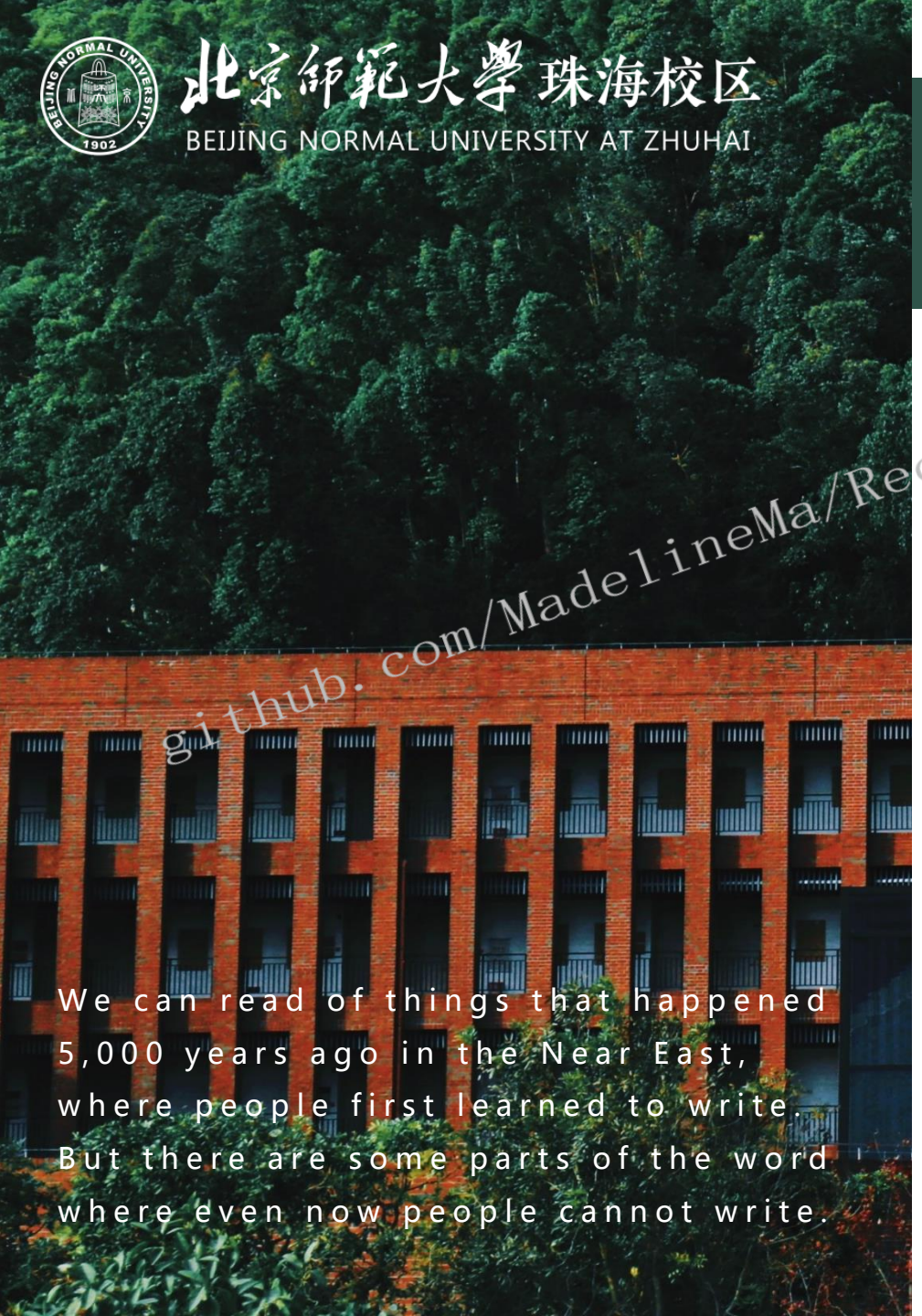

北京師範大學珠海校区
BEIJING NORMAL UNIVERSITY AT ZHUHAI
github.com/MadelineMa/Recommender-System
深度机器学习
知识补充
github.com/MadelineMa/Recommender-System
马静



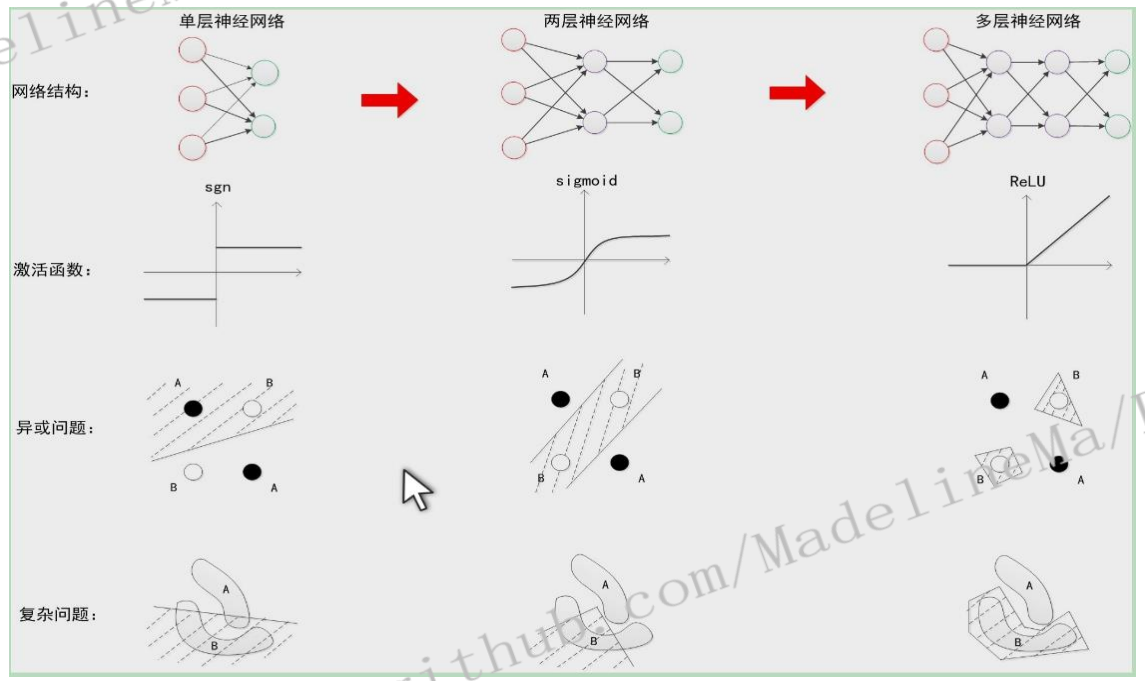
CONTENT

北京師範大學珠海校区
BEIJING NORMAL UNIVERSITY AT ZHUHAI

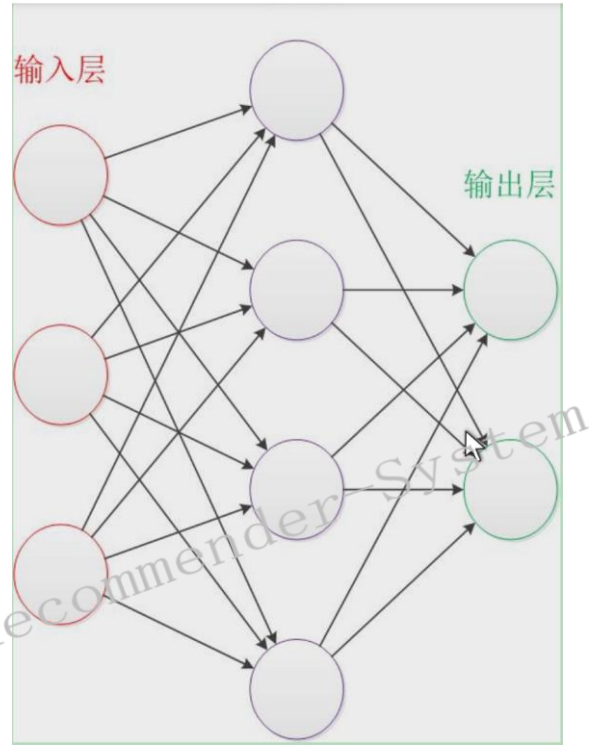
01NN概述

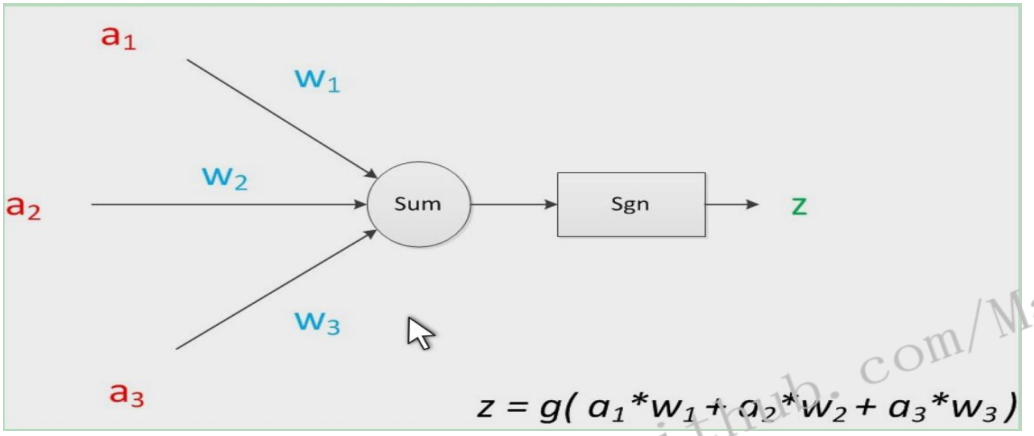
WIKI定义：一类通过多层非线性变换对高复杂性数据建模算法的合集.

证：只通过线性变换，任意层的全连接神经网络和单层神经网络模型对表达能力没有任何区别，而且都是线性模型.



- 设计一个神经网络时，输入层与输出层的节点数往往是固定的，中间层则可以自由指定；
- 神经网络结构图中的拓扑与箭头代表着预测过程时数据的流向，跟训练时的数据流有一定的区别；
- 结构图里的关键不是圆圈（代表“神经元”），而是连接线（代表“神经元”之间的连接）。每个连接线对应一个不同的权重（其值称为权值），这是需要训练得到的。





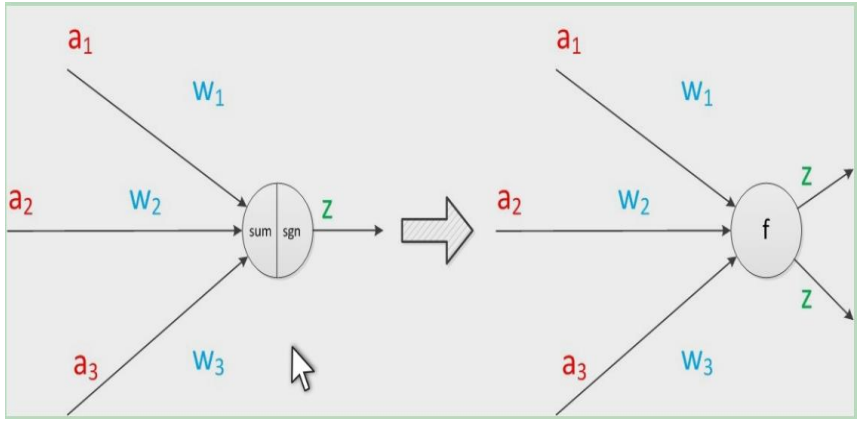
函数g是sgn函数，也就是取符号函数。这个函数当输入大于0时，输出1，否则输出0。

02
神经元

神经元

一个神经元可以引出多个代表输出的有向箭头，但值都是一样的；

神经元可以看作一个计算与存储单元。计算是神经元对其的输入进行计算功能。存储是神经元会暂存计算结果，并传递到下一层。



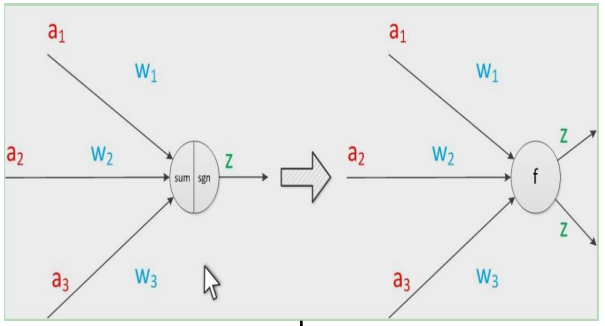
北京師範大學珠海校区
BEIJING NORMAL UNIVERSITY AT ZHUHAI

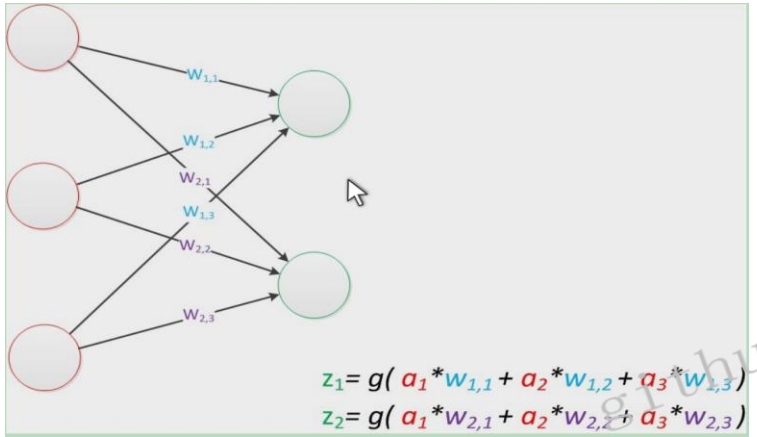
03 单层感知机

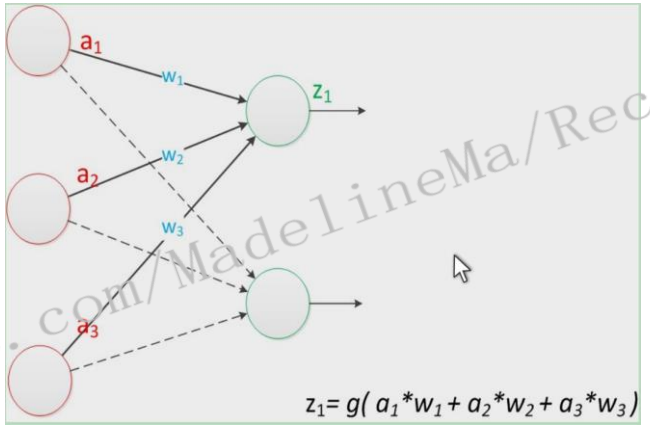
We can read of things that happened 5,000 years ago in the Near East, where people first learned to write.

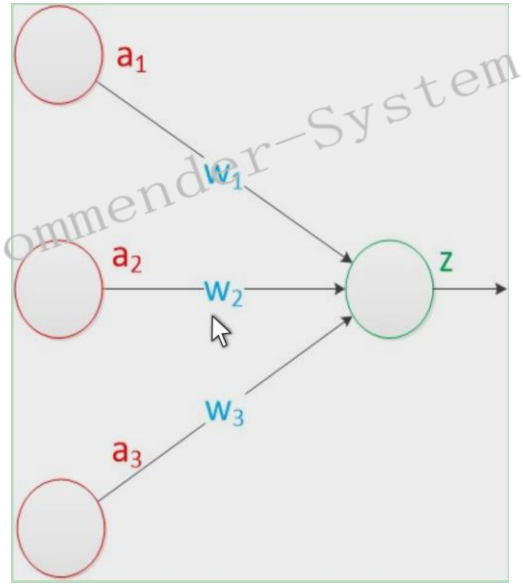
单层感知机

- 在原来神经元模型的“输入”位置添加神经元节点，标志其为“输入单元”；
- 在“感知器”中，有两个层次。分别是输入层和输出层。输入层里的“输入单元”只负责传输数据，不做计算。输出层里的“输出单元”则需要对前面一层的输入进行计算。
- 假如我们要预测的目标不再是一个值，而是一个向量，例如[2,3]。那么可以在输出层再增加一个“输出单元”。
- 计算的层次称之为“计算层”，并把拥有一个计算层的网络称之为“单层神经网络”。









$$g(W \cdot a) = z$$

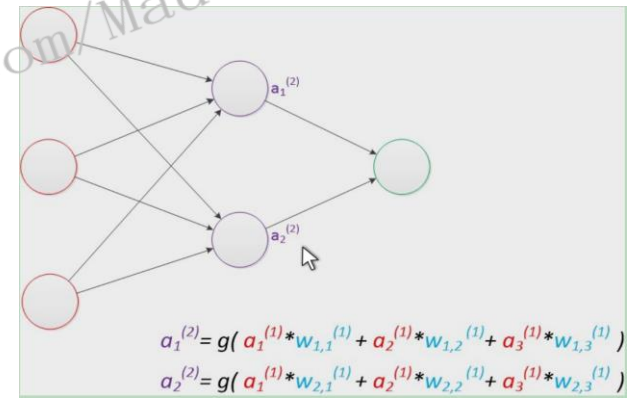
北京師範大學珠海校区
BEIJING NORMAL UNIVERSITY AT ZHUHAI

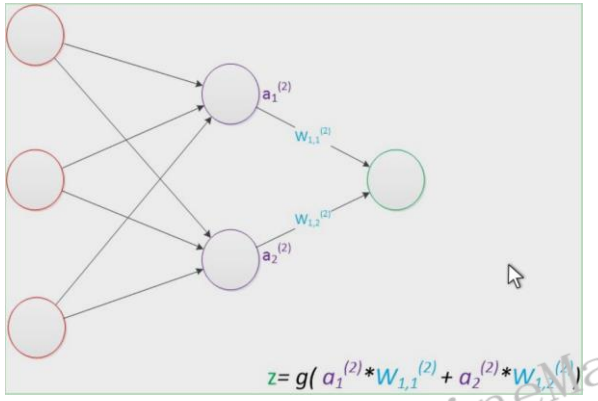
04
多层感知机

We can read of things that happened 5,000 years ago in the Near East, where people first learned to write.

多层感知机

- 两层神经网络除了包含一个输入层，一个输出层以外，还增加了一个中间层；
- 此时，中间层和输出层都是计算层；
- 权值矩阵增加到了两个，我们用上标来区分不同层次之间的变量.





$$g(W^{(1)} \cdot a^{(1)}) = a^{(2)}$$

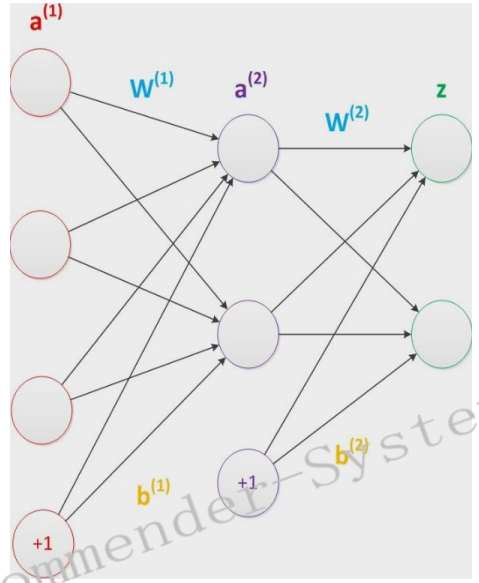
$$g(W^{(2)} \cdot a^{(2)}) = z$$

多层感知机

- 本质上是一个只含有存储功能，且存储值永远为1的单元；
- 在神经网络的每个层次中，除了输出层以外，都会含有这样一个偏置单元；
- 偏置节点很好认，因为其没有输入（前一层中没有箭头指向它）；
- 一般情况下，我们都不会明确画出偏置节点;

$$g\left(W^{(1)} \cdot a^{(1)} + b^{(1)}\right) = a^{(2)}$$

$$g\left(W^{(2)} \cdot a^{(2)} + b^{(1)}\right) = z$$



- 不再使用sgn函数作为函数g，而是使用函数sigmoid作为函数g;
- 类似“属于该类别的概率值”，可通过此值的大小进行对比和排序.

前向推导

对于Layer 2的输出 $a_1^{(2)}$ ， $a_2^{(2)}$ ， $a_3^{(2)}$ ，

$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma(w_{11}^{(2)} x_1 + w_{12}^{(2)} x_2 + w_{13}^{(2)} x_3 + b_1^{(2)})$$

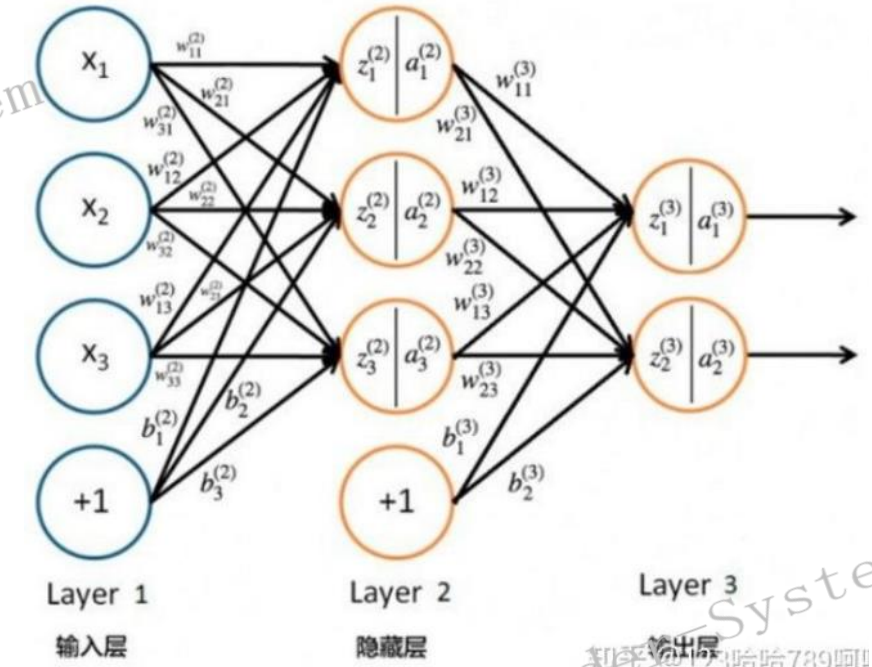
$$a_2^{(2)} = \sigma(z_2^{(2)}) = \sigma(w_{21}^{(2)} x_1 + w_{22}^{(2)} x_2 + w_{23}^{(2)} x_3 + b_2^{(2)})$$

$$a_3^{(2)} = \sigma(z_3^{(2)}) = \sigma(w_{31}^{(2)} x_1 + w_{32}^{(2)} x_2 + w_{33}^{(2)} x_3 + b_3^{(2)})$$

对于Layer 3的输出 $a_1^{(3)}$ ，

$$a_1^{(3)} = \sigma(z_1^{(3)}) = \sigma(w_{11}^{(3)} a_1^{(2)} + w_{12}^{(3)} a_2^{(2)} + w_{13}^{(3)} a_3^{(2)} + b_1^{(3)})$$

$$a_2^{(3)} = \sigma(z_2^{(3)}) = \sigma(w_{21}^{(3)} a_1^{(2)} + w_{22}^{(3)} a_2^{(2)} + w_{23}^{(3)} a_3^{(2)} + b_2^{(3)})$$



$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = \sigma(z^{(l)})$$

其中 σ 为 sigmoid 函数。

导数： $\sigma(z)(1-\sigma(z))$

简单介绍一下链式法则：

微积分中的链式法则（为了不与概率中的链式法则相混淆）用于计复合函数的导数。反向传播是一种计算链式法则的算法，使用高效的特定运输顺序。

设 x 是实数， f 和 g 是从实数映射到实数的函数。假设 $y = g(x)$ 并且 $z = f(g(x)) = f(y)$ 。那么链式法则就是： $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$ 。

前向推导

对于Layer 2的输出 $a_1^{(2)}$ ， $a_2^{(2)}$ ， $a_3^{(2)}$ ，

$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma(w_{11}^{(2)} x_1 + w_{12}^{(2)} x_2 + w_{13}^{(2)} x_3 + b_1^{(2)})$$

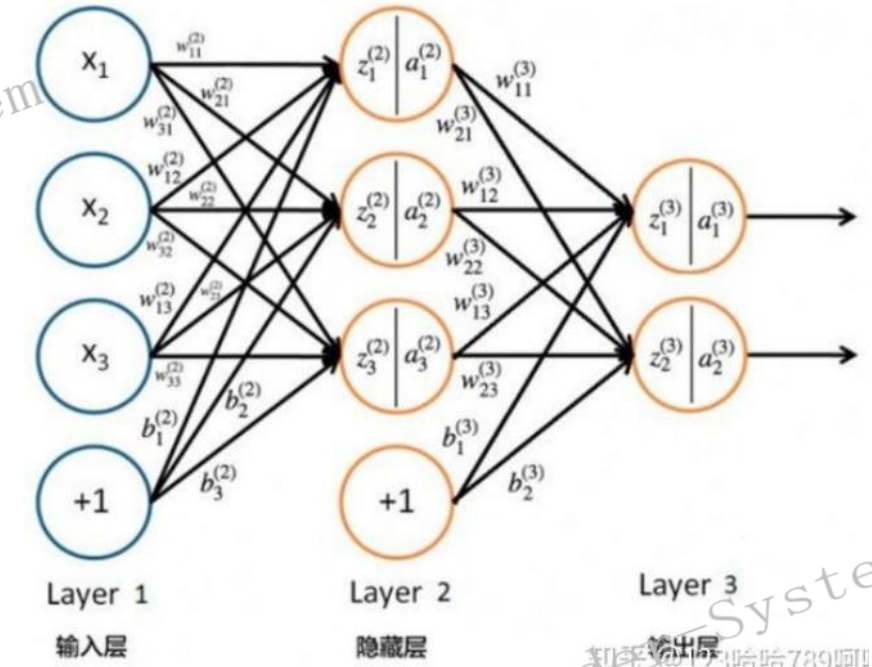
$$a_2^{(2)} = \sigma(z_2^{(2)}) = \sigma(w_{21}^{(2)} x_1 + w_{22}^{(2)} x_2 + w_{23}^{(2)} x_3 + b_2^{(2)})$$

$$a_3^{(2)} = \sigma(z_3^{(2)}) = \sigma(w_{31}^{(2)} x_1 + w_{32}^{(2)} x_2 + w_{33}^{(2)} x_3 + b_3^{(2)})$$

对于Layer 3的输出 $a_1^{(3)}$ ，

$$a_1^{(3)} = \sigma(z_1^{(3)}) = \sigma(w_{11}^{(3)} a_1^{(2)} + w_{12}^{(3)} a_2^{(2)} + w_{13}^{(3)} a_3^{(2)} + b_1^{(3)})$$

$$a_2^{(3)} = \sigma(z_2^{(3)}) = \sigma(w_{21}^{(3)} a_1^{(2)} + w_{22}^{(3)} a_2^{(2)} + w_{23}^{(3)} a_3^{(2)} + b_2^{(3)})$$



$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = \sigma(z^{(l)})$$

其中 σ 为 sigmoid 函数。

导数： $\sigma(z)(1-\sigma(z))$

输出层梯度

$$\frac{\partial C(W,b)}{\partial w^{(l)}} = \frac{\partial C(W,b)}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial w^{(l)}} = (a^{(l)} - y) \odot \sigma'(z^{(l)}) a^{(l-1)}$$

$$\frac{\partial C(W,b)}{\partial b^{(l)}} = \frac{\partial C(W,b)}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial b^{(l)}} = (a^{(l)} - y) \odot \sigma'(z^{(l)})$$

$$\delta^{(l)} = \frac{\partial C(W,b)}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} = (a^{(l)} - y) \odot \sigma'(z^{(l)})$$

前向推导

在进行反向传播算法前，我们需要选择一个损失函数，来度量训练样本计算出的输出和真实的训练样本输出之间的损失。我们使用最常见的**均方误差（MSE）来作为损失函数，**

$$C(W,b)=\frac{1}{2}\left\|a^{(l)}-y\right\|_2^2$$

$$z^{(l)}=W^{(l)}a^{(l-1)}+b^{(l)}$$

$$a^{(l)}=\sigma(z^{(l)})$$

其中 σ 为 sigmoid 函数。

输出层梯度

$$\frac{\partial C(W,b)}{\partial w^{(l)}}=\frac{\partial C(W,b)}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial w^{(l)}}$$
$$=(a^{(l)}-y)\odot\sigma'(z^{(l)})a^{(l-1)}$$

$$\frac{\partial C(W,b)}{\partial b^{(l)}}=\frac{\partial C(W,b)}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial b^{(l)}}$$
$$=(a^{(l)}-y)\odot\sigma'(z^{(l)})$$

正向推导记录

激活函数求导

每层损失

方便推导，令：$\delta^{(l)}=\frac{\partial C(W,b)}{\partial a^{(l)}}\frac{\partial a^{(l)}}{\partial z^{(l)}}=(a^{(l)}-y)\odot\sigma'(z^{(l)})$

反向递推

这里我们用数学归纳法，假设第 $l+1$ 层的 $\delta^{(l+1)}$ 已经求出，那么我们如何求出第 l 层的 $\delta^{(l)}$ 呢?

$$\delta^{(l)} = \frac{\partial C(W,b)}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}} = \delta^{(l+1)} \frac{\partial z^{(l+1)}}{\partial z^{(l)}}$$

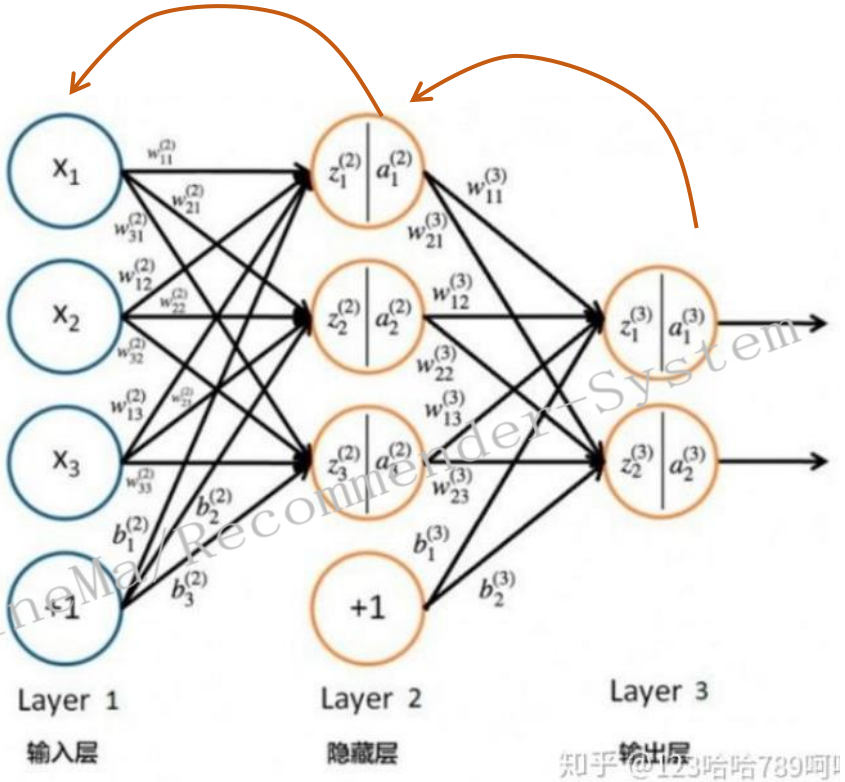
而 $z^{(l+1)}$ 和 $z^{(l)}$ 的关系如下:

$$z^{(l+1)} = W^{(l+1)} a^{(l)} + b^{(l+1)} = W^{(l+1)} \sigma(z^{(l)}) + b^{(l+1)}$$

这样很容易求出，

$$\frac{\partial z^{(l+1)}}{\partial z^{(l)}} = (W^{(l+1)})^T \odot \sigma'(z^{(l)})$$

$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(z^{(l)})$$



反向递推

$$\delta^{(l)} = \frac{\partial C(W,b)}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}} = \delta^{(l+1)} \frac{\partial z^{(l+1)}}{\partial z^{(l)}} \qquad \frac{\partial z^{(l+1)}}{\partial z^{(l)}} = (W^{(l+1)})^T \odot \sigma'(z^{(l)})$$

$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(z^{(l)})$$

现在我们得到了 $\delta^{(l)}$ 的递推关系式，只要求出了某一层的 $\delta^{(l)}$，求解 $w^{(l)}$，$b^{(l)}$ 的对应梯度就很简单：

$$\frac{\partial C(W,b)}{\partial w^{(l)}} = \delta^{(l)} a^{(l-1)}$$

$$\frac{\partial C(W,b)}{\partial b^{(l)}} = \delta^{(l)}$$

反向递推

$$\delta^{(l)} = \frac{\partial C(W,b)}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}} = \delta^{(l+1)} \frac{\partial z^{(l+1)}}{\partial z^{(l)}} \qquad \frac{\partial z^{(l+1)}}{\partial z^{(l)}} = (W^{(l+1)})^T \odot \sigma'(z^{(l)})$$

$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(z^{(l)})$$

现在我们得到了 $\delta^{(l)}$ 的递推关系式，只要求出了某一层的 $\delta^{(l)}$，求解 $w^{(l)}$，$b^{(l)}$ 的对应梯度就很简单：

$$\frac{\partial C(W,b)}{\partial w^{(l)}} = \delta^{(l)} a^{(l-1)}$$

$$\frac{\partial C(W,b)}{\partial b^{(l)}} = \delta^{(l)}$$

流程

1. 初始化参数W，b

2. 进行前向传播算法计算，for $l = 2$ to L

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = \sigma(z^{(l)})$$

3. 通过损失函数计算输出层的梯度

4. 进行反向传播算法计算，for $l = L - 1$ to 2

$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(z^{(l)})$$

5. 更新W，b

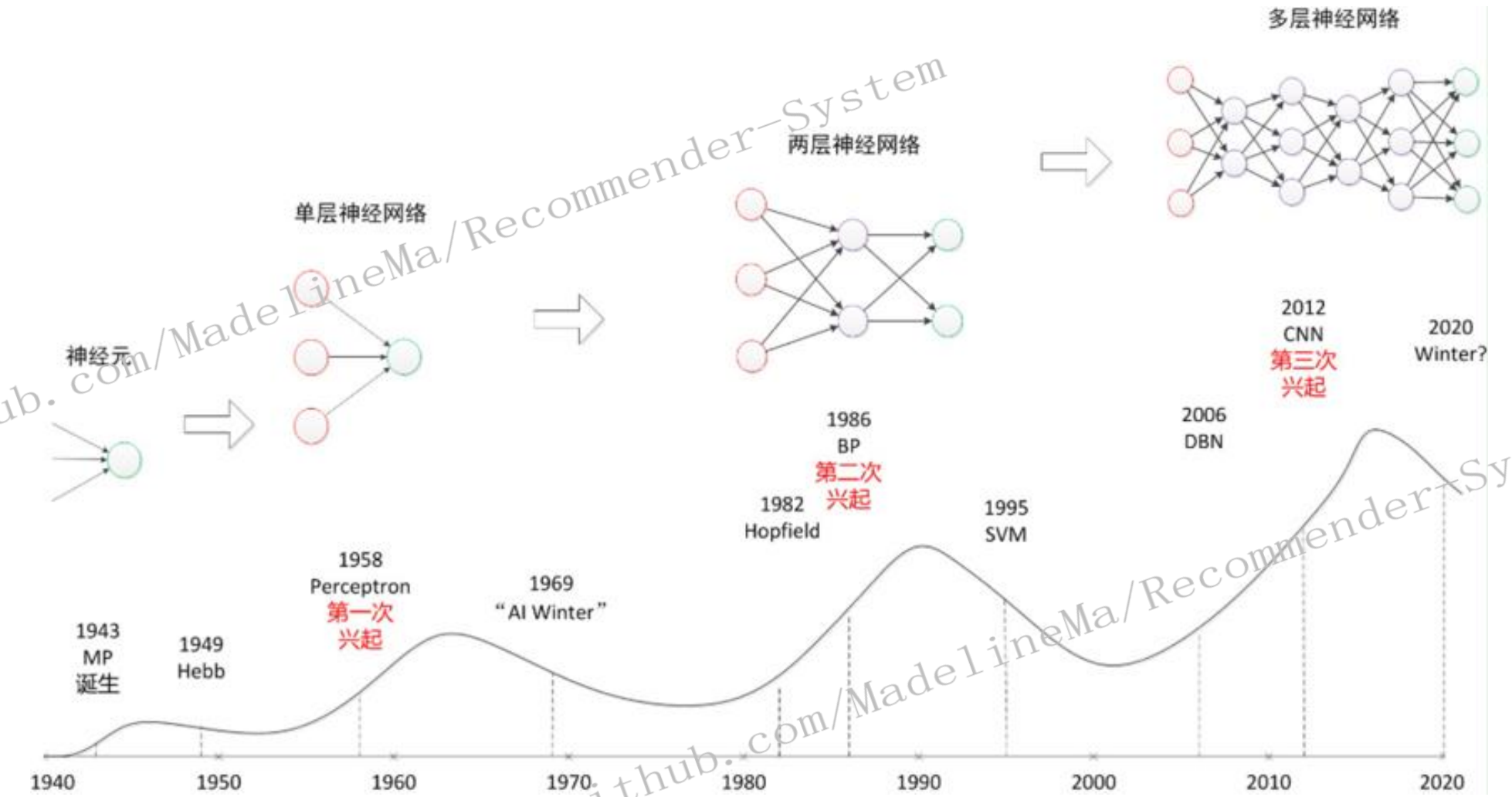
通过梯度下降算法更新权重 w 和偏置 b 的值，α 为学习率其中 $\alpha \in (0, 1]$。

$$w^{(l)} = w^{(l)} - \alpha \frac{\partial C(w,b)}{\partial w^{(l)}}$$

$$b^{(l)} = b^{(l)} - \alpha \frac{\partial C(w,b)}{\partial b^{(l)}}$$

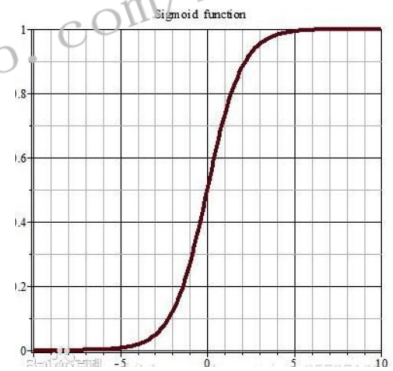
6. 如果所有W，b的变化值都小于停止迭代阈值ϵ，则跳出迭代循环

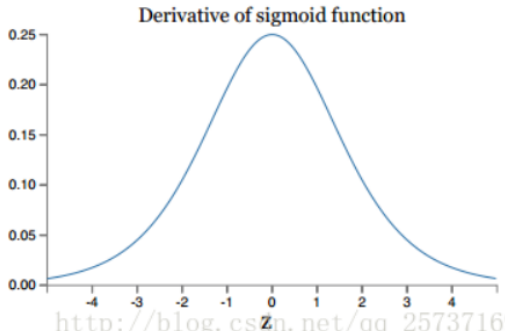
7. 输出各隐藏层与输出层的线性关系系数矩阵W和偏置b。

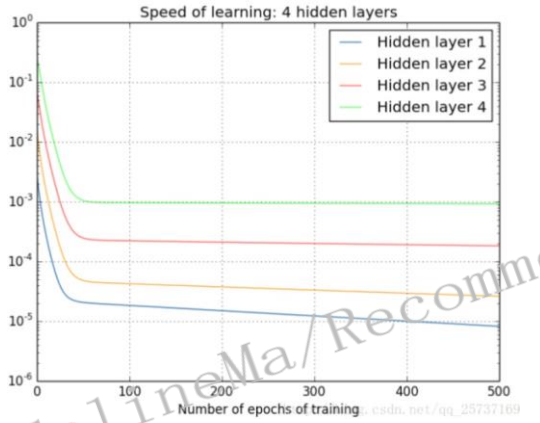


反向传播算法的启示是数学中的链式法则。所以在传播过程中需要注意梯度消失和梯度爆炸等问题。

对激活函数进行求导，如果此部分大于1，那么层数增多的时候，最终的求出的梯度更新将以指数形式增加，即发生**梯度爆炸**，如果此部分小于1，那么随着层数增多，求出的梯度更新信息将会以指数形式衰减，即发生了**梯度消失**。





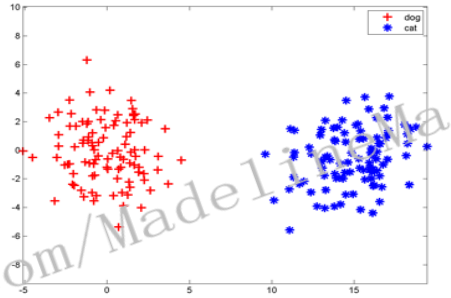


Sigmoid导数不超过0.25，所以易发生梯度消失，以及远离输出的隐藏层更新速度慢。

面试常见题：梯度消失和梯度爆炸解决方法.

- 机器学习问题之所以称为学习问题，而不是优化问题，就是因为它不仅要求数据在训练集上求得一个较小的误差，在测试集上也要表现好。
- 提升模型在测试集上的预测效果的主题叫做泛化（generalization），相关方法被称作正则化（regularization）。
- 面试常见题：防止overfitting的方法？

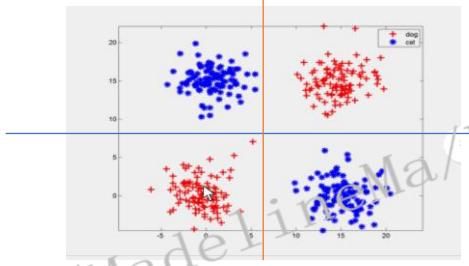
假设分类器的输入是通过某种途径获得的数值，输出是0和1，比如分别代表猫和狗。现在有一些样本：

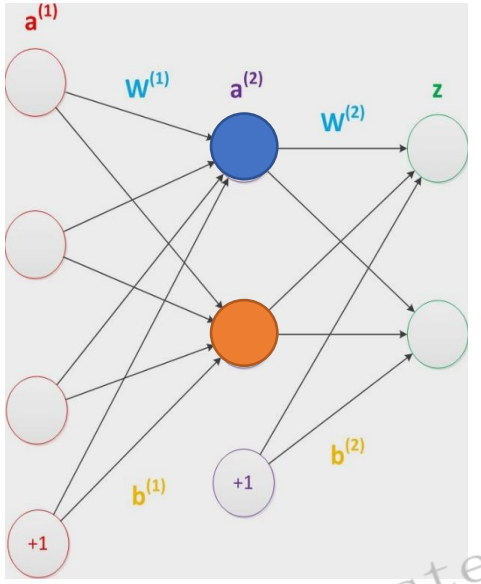


一条直线把平面一分为二，一个平面把三维空间一分为二，一个n-1维超平面把n维空间一分为二，两边分属不同的两类，这种分类器就叫做神经元。

- 神经元模型的一种学习方法称为**Hebb算法**：先随机选一条直线/平面/超平面，然后把样本一个个拿过来，如果这条直线分错了，说明这个点分错边了，就稍微把直线移动一点，让它靠近这个样本，争取跨过这个样本，让它跑到直线正确的一侧；如果直线分对了，它就暂时停下不动。因此训练神经元的过程就是这条直线不断在跳舞，最终跳到两个类之间的竖直线位置。

- 神经元的一个缺点是：它只能切一刀！那么下面两个类要怎么区分？

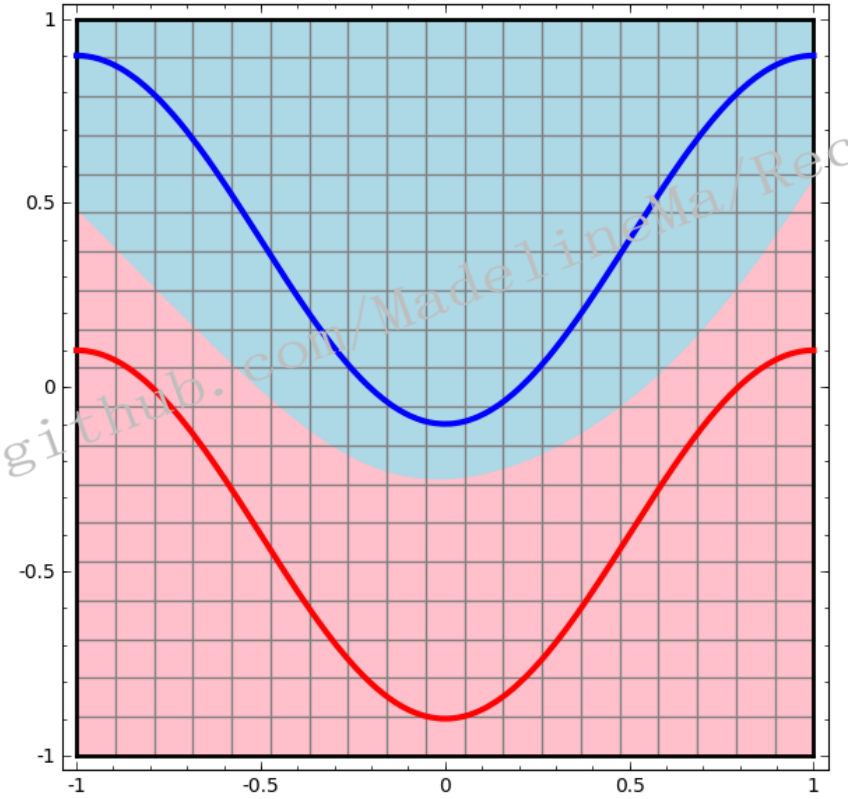




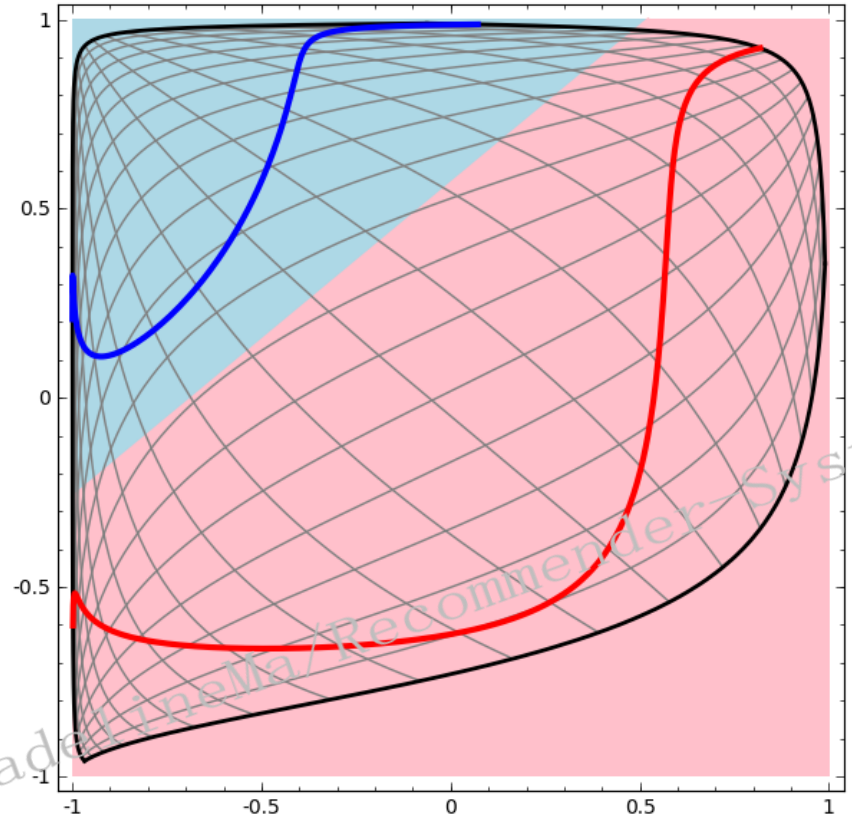
- 解决办法是多层神经网络，底层神经元的输出是高层神经元的输入。我们可以在中间横着砍一刀，竖着砍一刀，然后把左上和右下的部分合在一起，与右上的左下部分分开；也可以围着左上角的边沿砍10刀把这一部分先挖出来，然后和右下角合并。

- **每砍一刀，其实就是使用了一个神经元，把不同砍下的半平面做交、并等运算，就是把这些神经元的输出当作输入，后面再连接一个神经元。**这个例子中特征的形状称为异或，这种情况一个神经元搞不定，但是两层神经元就能正确对其进行分类。

理论证明，两层神经网络可以无限逼近任意连续函数



两层神经网络决策分界



输出层决策分界

单层网络只能做线性分类任务。而两层神经网络中的后一层也是线性分类层，应该只能做线性分类任务。

本周作业

1. 自行推导正向、反向递推过程
2. 查找两个面试题的答案
3. 依据本节基础，可开始自行学习CNN，RNN

Reference：

详解深度学习中的梯度消失、爆炸原因及其解决方法 - 知乎 (zhihu.com)

超详细的Git使用教程(图文)_kinggm的博客-CSDN博客_git使用教程


北京師範大學珠海校区
BEIJING NORMAL UNIVERSITY AT ZHUHAI
github.com/MadelineMa/Recommender-System
THANKS
github.com/MadelineMa/Recommender-System
DESIGNED BY 2xh