Extra：归一化详解

归一化的“一”
到底是什么意思

1号小哥哥
特征：**1.7**；**4700**

隐含了单位 →

1.7（米）
4700（元/月）
单位→基本属性

不能直接 →
发生关系

量纲

长度-“L”；单位-m
时间-“T”；单位-s
L/T导出速度（m/s）

↓

核对$\boldsymbol{X_{norm}}$的量纲

$$X_{norm} = \frac{(m)}{(m)} = 1$$

纯数量纲？

单位“消失”了

答案：量纲转化为1，去除影响

$$X_{norm} = \frac{X - \mu}{\sigma}$$

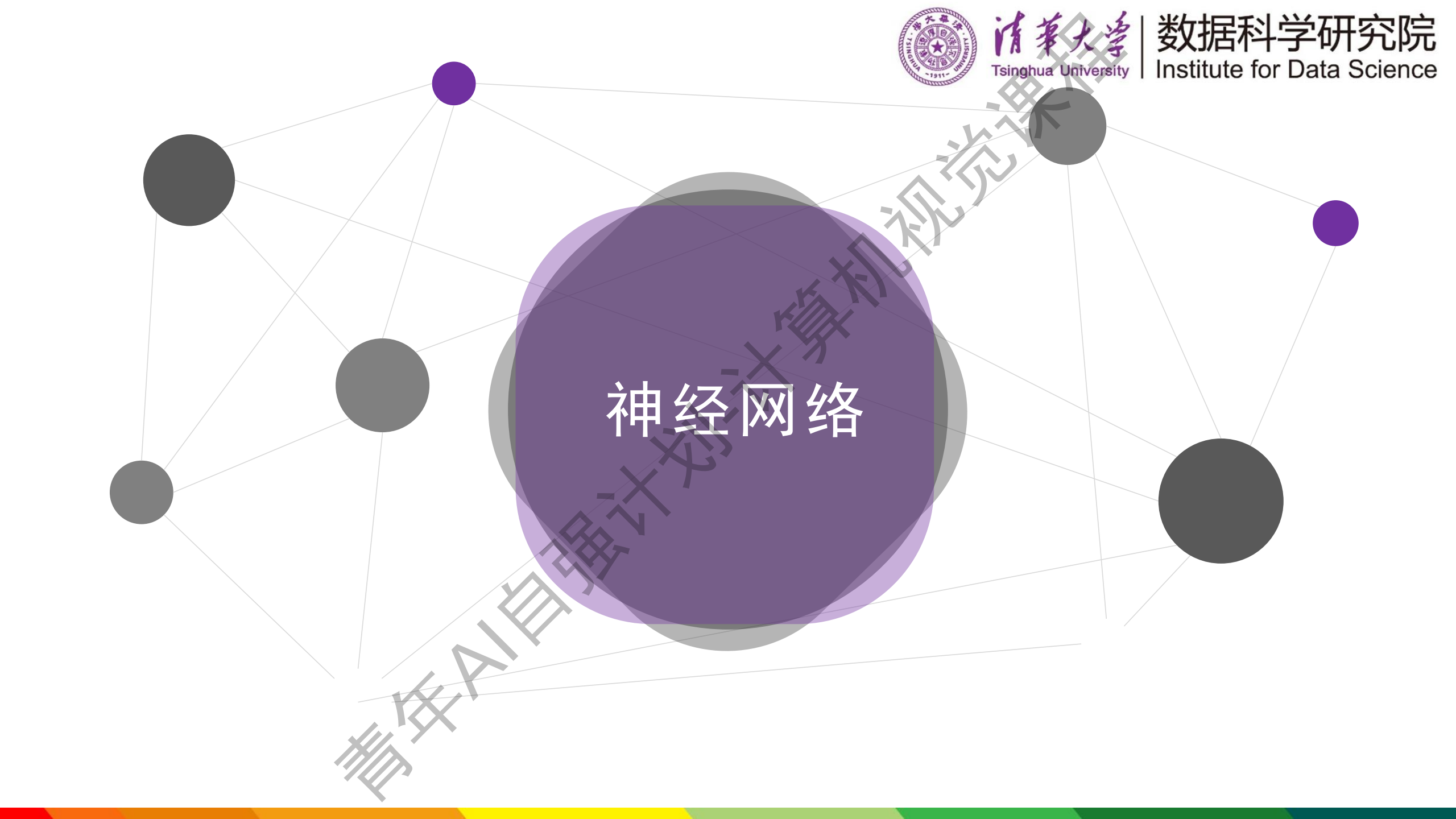
归一化为mu=0; Sigma=1
不一定必为[-1,1]

$$X_{norm} = \frac{X - \mu}{X_{max} - X_{min}}$$

归一化到 [-1,1]

$$X_{norm} = \frac{2(X - X_{min})}{X_{max} - X_{min}} - 1$$

归一化到 [-1,1]


清华大学
Tsinghua University
数据科学研究院
Institute for Data Science
神经网络
青年AI自强计划计算机视觉课程

0.1：重要更新

学堂在线 xuetangx.com

本课程正式上线：
搜索“青年AI”即可关注

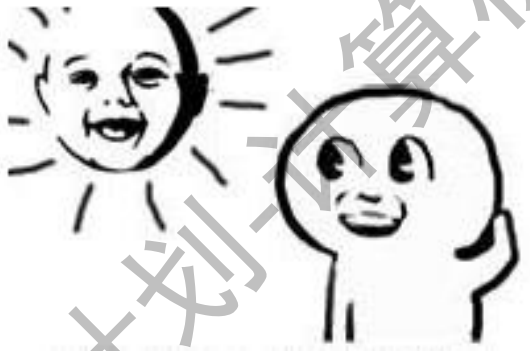
资料发布、直播、作业回收等
主要依托学堂在线

0.2：明确课程定位

垂直行业从业者&爱好者

垂直行业从业者&爱好者：
听课目标：0→0.5
定性理解，专注落地
提升办法：联系我们

算法科学家



算法科学家：
听课目标：0→1
初步入门，加强算法
提升办法：共享论文

AI工程师

AI工程师：
听课目标：0→1
初步入门，加强代码
提升办法：工程问题

1.1：背景带入

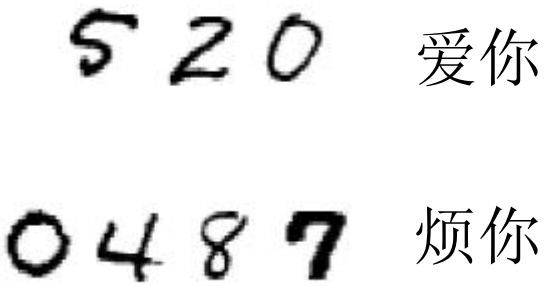
可还记得
数据院小姐姐？

立志找到
全局最优解

世纪佳缘
1.7亿注册用户
初步筛选

浪漫（zuosi）
小要求

手写数字表达心意



小问题

向3000万发候选者出了信息

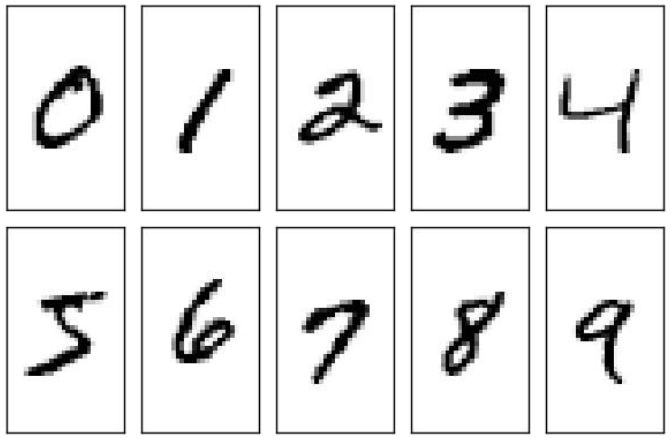
小需求

训练
“识别手写数字”
的算法

1.2：数据集简介

是否有合适的数据集？
坏消息&好消息

MNIST



基本信息

训练集：60,000 个样本

测试集：10,000 个样本

尺寸：28 * 28 像素

格式：灰度图像

train-images-idx3-ubyte.gz:	training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:	training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:	test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:	test set labels (4542 bytes)

大家可以到官网免费下载：http://yann.lecun.com/exdb/mnist/

N听着很厉害
为啥是M？

M=modified

2.1：OVA方法介绍

小任务/作业1：
用LR拟合MNIST

LR是个二分类的分类器，
这是个10分类，怎么破？

One vs All

“一对多”的真正含义
不是1个分类器“打”10个类别;
而是1个分类器只“打”1个类别，要想解决10分类的问题，就训练10个分类器好了。

Reference

[1] Rifkin R, Klautau A. In defense of one-vs-all classification[J]. Journal of machine learning research, 2004, 5(Jan): 101-141.

2.2：训练过程

What We See

人类看到的

What Computers See

计算机看到的

图片对于计算机是什么？

MNIST为图片28*28灰度图

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ \cdots & \cdots \\ x_1^{(50)} & x_2^{(50)} \end{bmatrix}$$

上堂课的X

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_{784}^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_{784}^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{(60,000)} & x_2^{(60,000)} & \cdots & x_{784}^{(60,000)} \end{bmatrix}$$

本堂课的X

2.3：提出问题

$$Y=\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(60,000)} \end{bmatrix}$$

$$\theta^T X = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \dots + \theta_{784} x_{784}$$

$$h_\theta(x) = g(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

$$\theta_j = \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j}$$

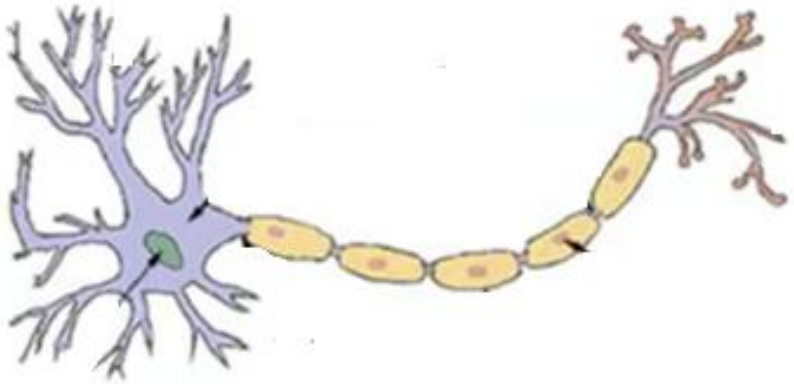
超参配置	手写字-数字	精度·/%
训练样本数：55000 Batchsize：30 迭代次数：1000000 Lr:learning·rate·decay，0.001 优化算法：梯度下降 初始化：0.0 Loss·function：交叉熵	0	55.34
	1	53.65
	2	56.14
	3	55.47
	4	57.18
	5	62.20
	6	58.84
	7	54.49
	8	59.30
	9	58.08

精度50%还不错？

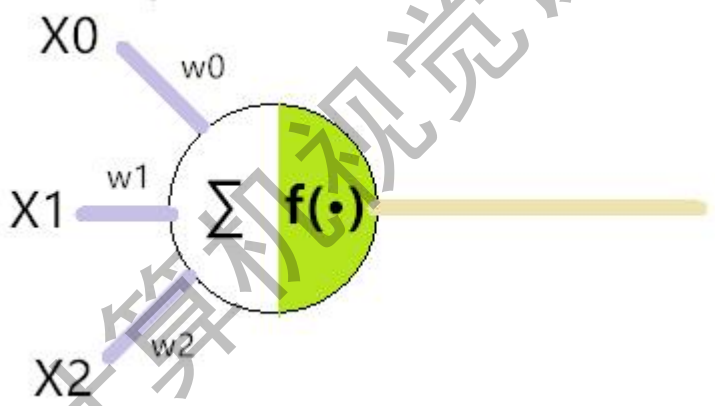
超参配置	手写字-数字	精度·/%
训练样本数：500 迭代次数：1000（已稳定） Lr:0.01 优化算法：梯度下降 初始化：0.001 Loss·function：交叉熵	0	87.6
	1	89.9
	2	90.6
	3	88.6
	4	90.4
	5	92.6
	6	90.4
	7	89.6
	8	89.8
	9	90.6

这次精度为啥是90%

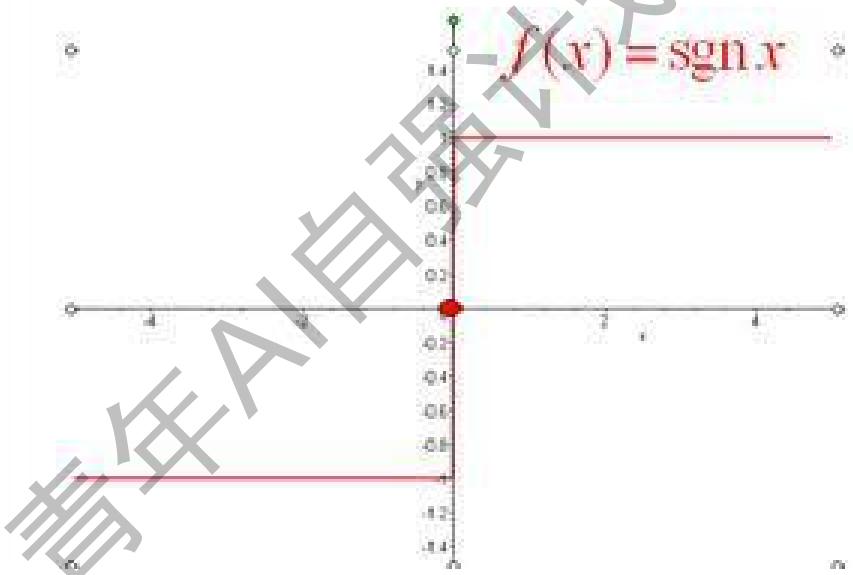
3.1：NN的历史渊源



生理学上的神经元

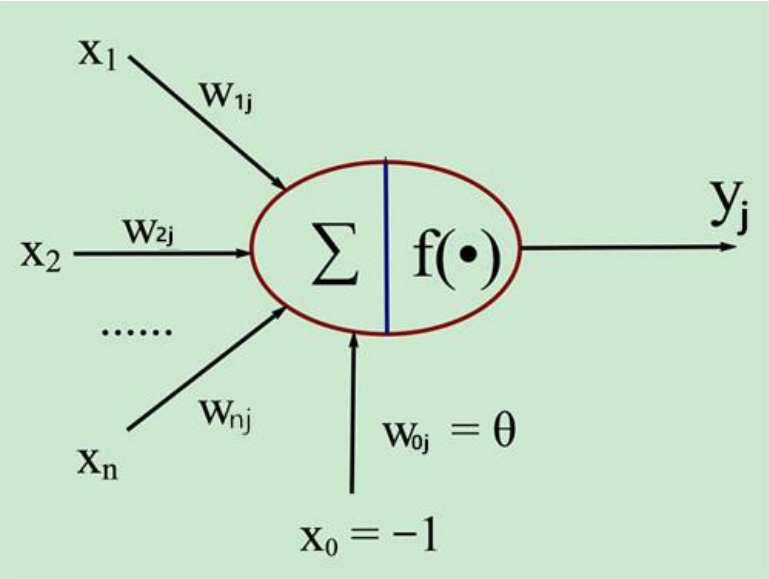


数学模型仿生

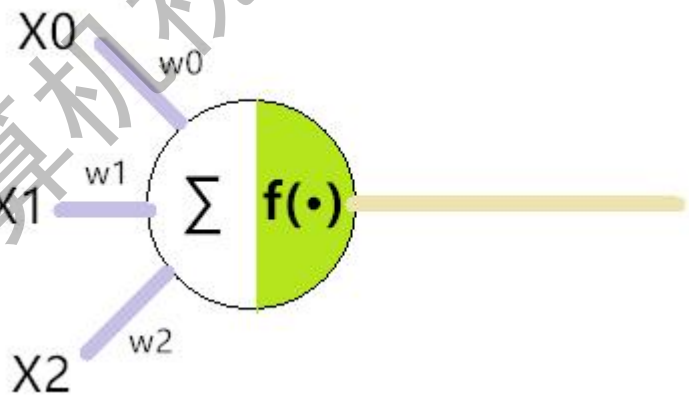


体会“激活”的含义

3.1：NN的历史渊源



“神经元”早期的数学模型

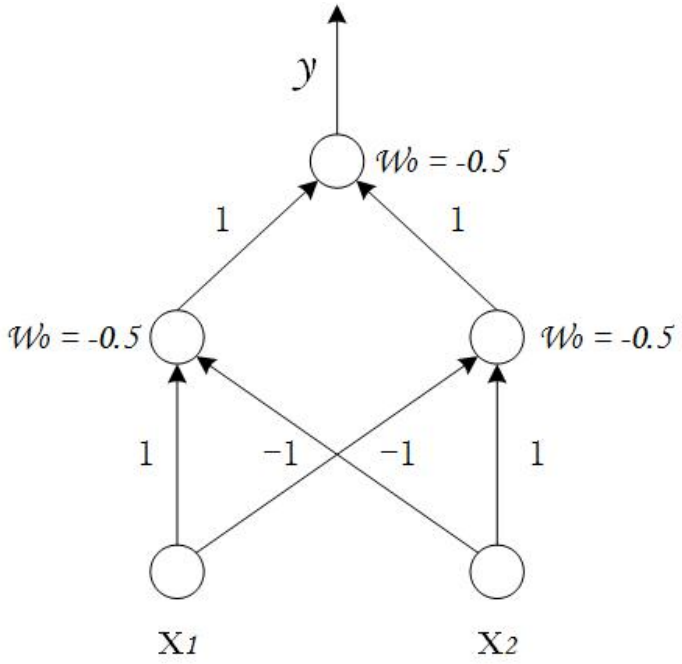


写出数学公式

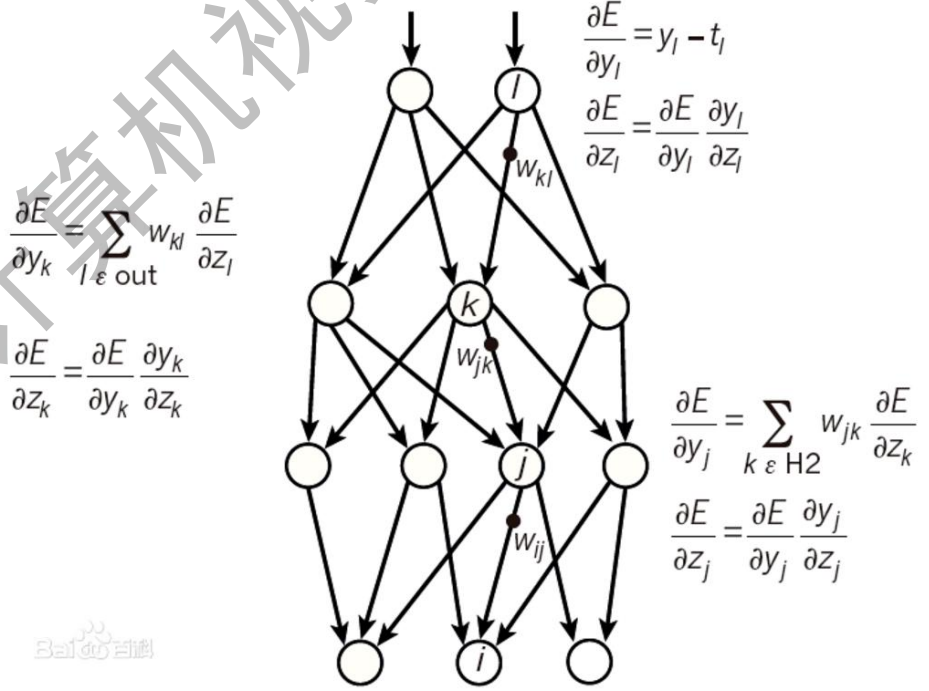
$$f(w_0 + w_1x_1 + w_2x_2)$$

$$g(\theta_0 + \theta_1x_1 + \theta_2x_2)$$

3.1：NN的历史渊源



多个神经元
感知机1958年



BP神经网络
Hinton1986年

3.2：模型结构及前馈传播

简化问题
有具象参考

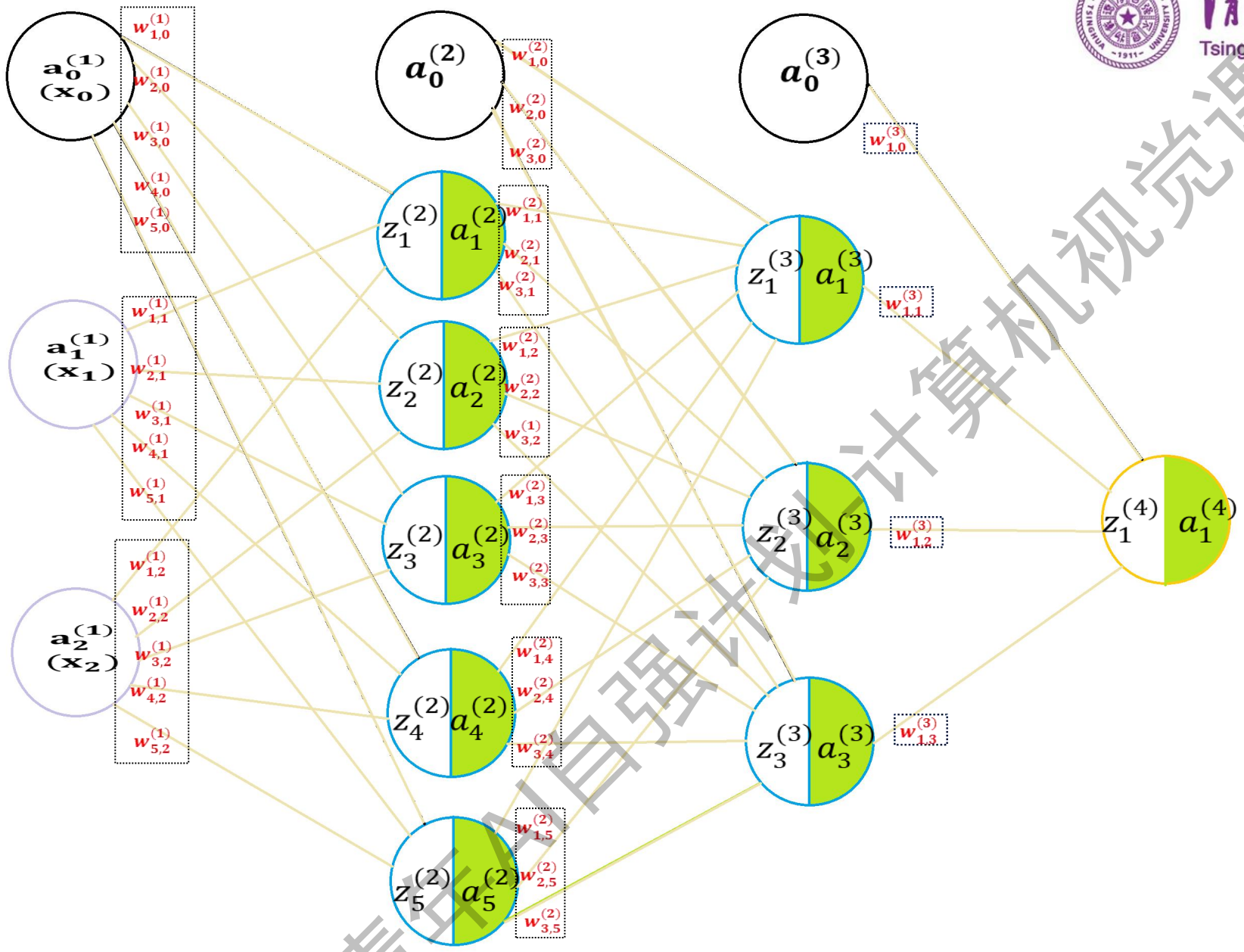
用NN拟合
上节课数据集

模型结构

参数更新

预防针：

1、没有任何捷径；
2、仔细弄懂每个符号，每个运算过程；
3、有师兄带你仔细梳理，系统总结



前馈传播示意图

3.2：FP（forward propagation）结构总述

何谓“前”馈

神经网络是一个仿生数学模型，我们的变量x代表神经系统的“输入信号”
前馈传播，是要把输入信号，从输入层（图中的Layer1）逐层“向前”传播到整个网络的最后一层（图中的Layer4）

结构：

1. 我们图中用1个彩色小圆圈表示一个神经元（上方黑色圆圈表示“偏置”（bias）输入，不是神经元）

2. 偏置（bias）是额外输入的，他是一个所有元素都为“1”的单位向量（identity vector），其作用可以参考“常数项”在“一次函数”中的作用（让函数图像离开原点）

3. 本例中的网络有4层，信号从第1层神经元（用淡紫色画出）输入，在第4层神经元（用亮黄色画出）输出。第1 层和第4层的神经元个数是确定的，具体参见下面一条

4. 我们选取了多少特征，输入层就有多少神经元；我们的目标是“几分类”，输出层就有几个神经元

3.2：FP-结构总述

结构：

5、除了第1层和最后一层之外，其他都被称作隐藏层（蓝色圆圈画出），隐藏层的结构是我们手动设置的，如在本例中，采用了两个隐藏层的设置，分别是图中的第2层和第3层。其中我们为第2层设置了5个神经元，并为第3层设置了3个神经元

6、图中每一条淡黄色的直线，代表一个“权值”（weight），用w表示。（这是我们模型的主要参数，相当于上一堂课的θ。采用w代表参数一方面是为了尊重原文，另一方面也是为了编程方便）注意：我们训练模型的目标就是更改这些w代表的参数，来使损失函数最小

7、层与层之间神经元是全连接，即下一层有几个神经元，本层的每个神经元就会有几条线发出并与下一层的每个神经元相连

3.2：FP-运算过程

权值用法

所有浅层的“小圆圈”（一般代表“神经元”）都通过一条代表权值的线，与下一层的小圆圈相连。这表示的意思是：将浅层神经元（小圆圈）里装载的变量，乘以对应的权值（w，黄色的线），输入到下一层神经元中。

运算：

大家会发现一个问题，除了第1层外，其它层的神经元内部都分成了两部分，分别是：

1. 用白色填充代表的“加和运算”部分，在这里，神经元将上一层传来的信号（乘以了权值的各上层变量），简单的全部相加起来；
2. 用绿色填充代表的“激活运算”部分。在这里我们将做过加和的信号，输入一个“激活函数”（本例中为sigmoid函数），并将得到的输出继续下在下一层传播。
3. 如此往复，直到最后的输出层。输出层有几个神经元，就意味着我们训练了几个分类器，由于本例中输出层所采用的激活函数同样为sigmoid，所以每一个神经元的输出信号都直接可以用来表示各自代表分类器的预测结果。

3.2：FP-符号释义

概述

把图中出现的所有字母类符号分为两类，一类是代表输入信号的变量（用黑色字体写出）,具体为：x、z、a;。
另一类是模型本身的参数（用红色字体写出），只有w。从图中可以发现w的数量非常多，但是规律性特别强。。

详解：

1. 神经元中的x代表输入的样本特征，样本有几个维度的特征，输入层就有几个神经元。本例中我们为小哥哥选择了“身高”和“月收入”两个特征，所以输入层有2个神经元。
2. 图中的z表示当前神经元接收到上游传来的信号后，做完加和运算的结果；
3. 图中的a表示同一个神经元中的z，经过激活函数（本例中为sigmoid）后的结果；

3.2：FP-符号释义

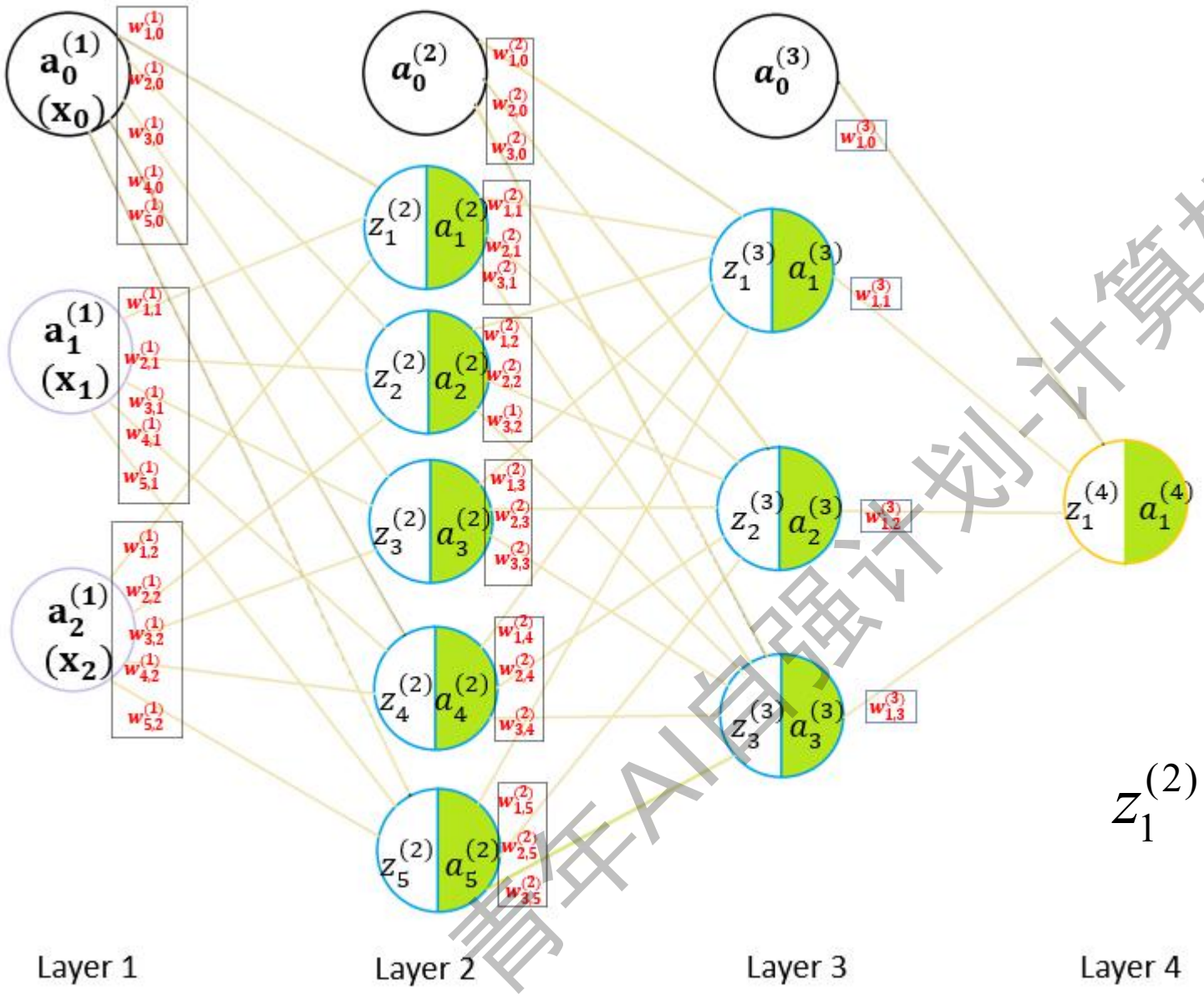
详解：

4、几乎所有的符号都同时拥有上角标和下角标。所有符号的上角标，都表示该符号所属的层数，用l来表示。如a^l表示第l层中所有神经元激活后的传输；

5、对于a_j^l和z_j^l，下角标只有1位记为j，表示其所属神经元在其所属层的序号，如a_1^l表示第l层的第1个神经元的输出；

6、对于$w_{i,j}^l$下角标有2位，我们说每一个$w_{i,j}^l$都是为了链接第l层和第$l+1$层的两个不同神经元，信号从第l层，第j个神经元出发，乘以$w_{i,j}^l$后，传输到第$l+1$层，第i个神经元处；

3.2：FP-尝试计算



尝试写出 $a_1^{(2)}$

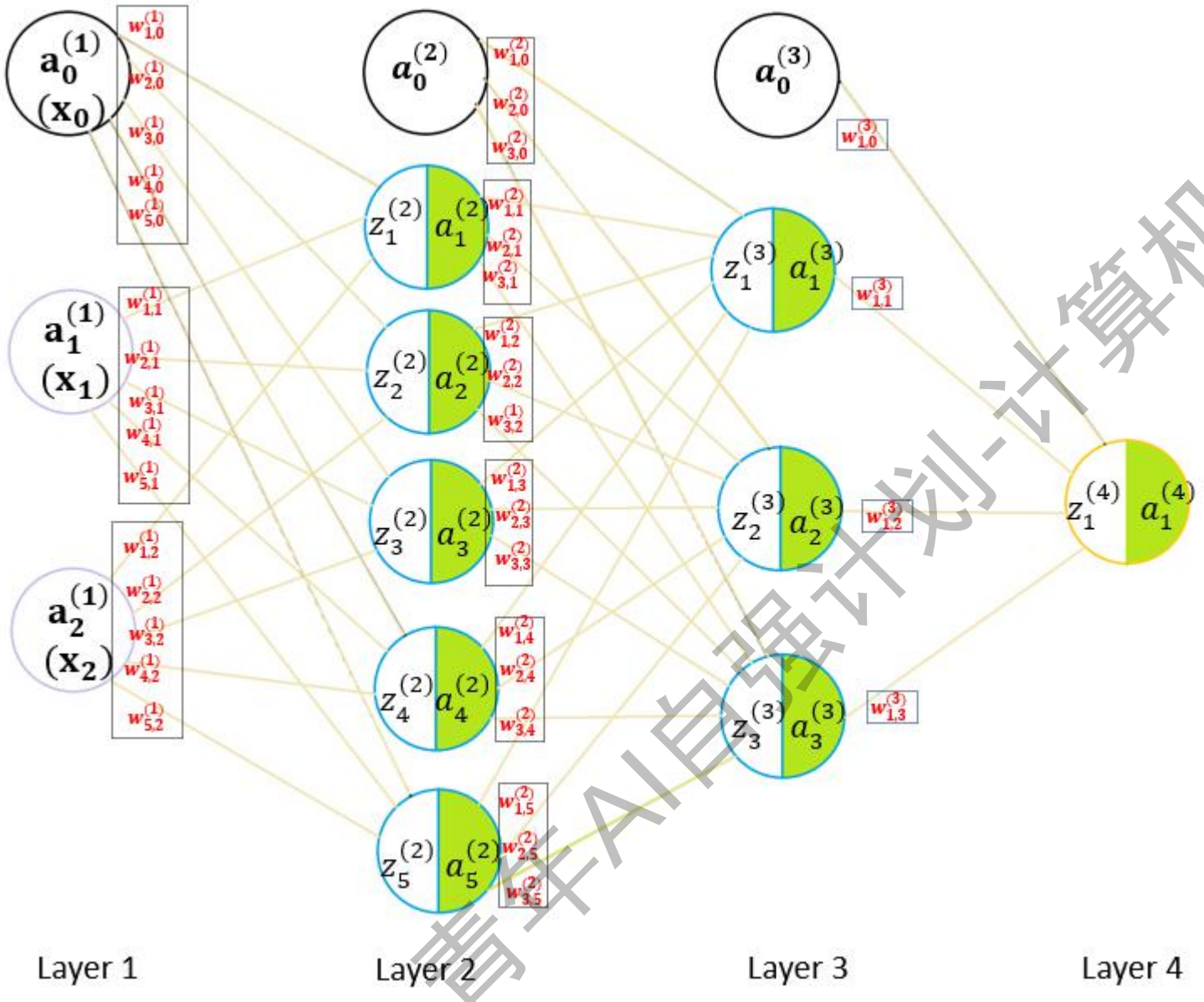
找到神经元，先写z

$$a_1^{(2)} = g(z_1^{(2)})$$

找到对应的权值

$$z_1^{(2)} = w_{1,0}^{(1)} a_0^{(1)} + w_{1,1}^{(1)} a_1^{(1)} + w_{1,2}^{(1)} a_2^{(1)}$$

3.2：FP- 矩阵化



给出矩阵化公式

$$a^{(1)} = x$$

$$z^{(2)} = w^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

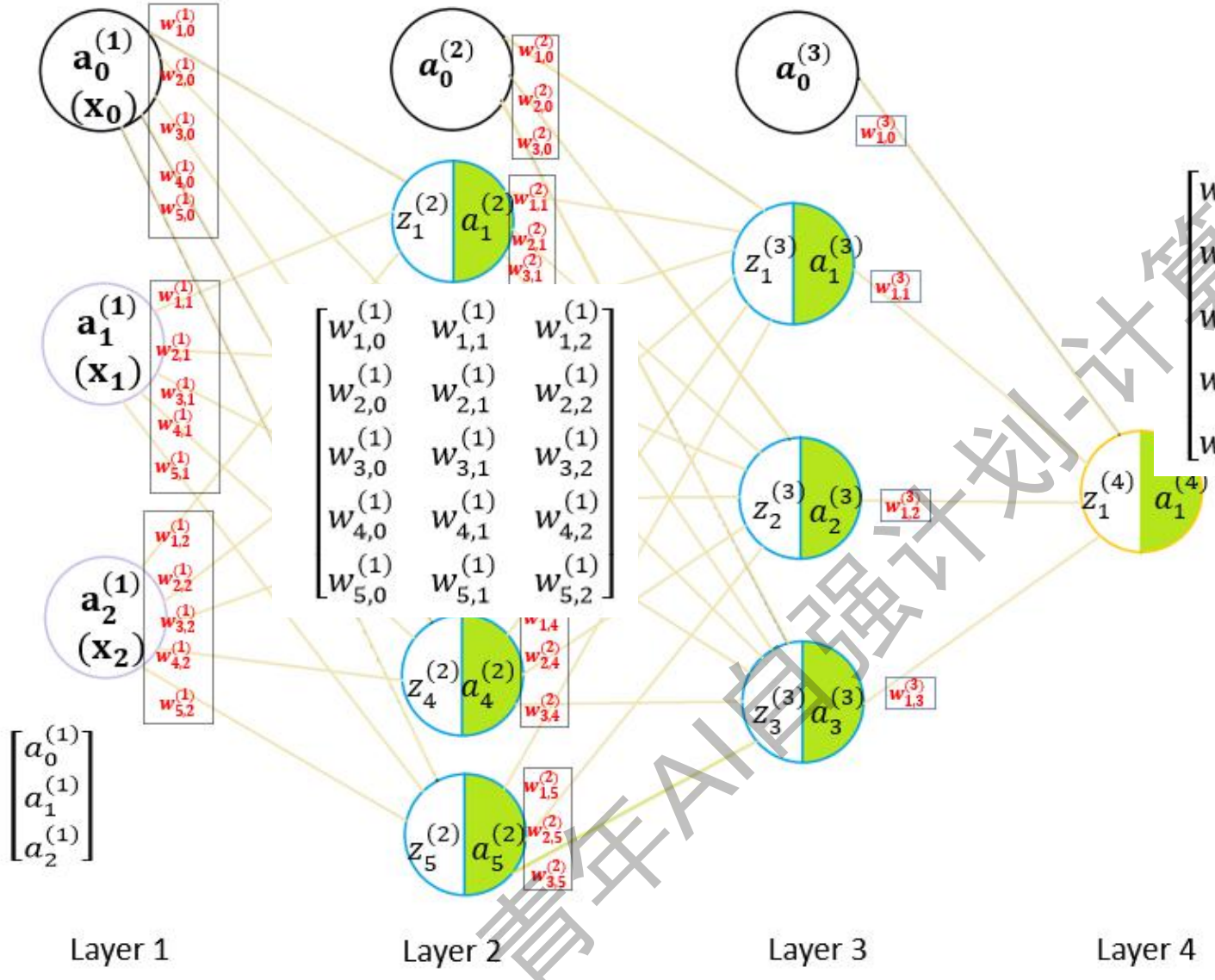
$$z^{(3)} = w^{(2)} a^{(2)}$$

$$a^{(3)} = g(z^{(3)})$$

$$z^{(4)} = w^{(3)} a^{(3)}$$

$$a^{(4)} = g(z^{(4)})$$

3.2：FP- 矩阵化



验证矩阵化公式：求$z_1^{(2)}$

$$\begin{bmatrix} w_{1,0}^{(1)} & w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ w_{2,0}^{(1)} & w_{2,1}^{(1)} & w_{2,2}^{(1)} \\ w_{3,0}^{(1)} & w_{3,1}^{(1)} & w_{3,2}^{(1)} \\ w_{4,0}^{(1)} & w_{4,1}^{(1)} & w_{4,2}^{(1)} \\ w_{5,0}^{(1)} & w_{5,1}^{(1)} & w_{5,2}^{(1)} \end{bmatrix} \times \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \\ z_5^{(2)} \end{bmatrix}$$

规则：依次相乘再相加

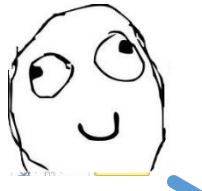
"Dot Product" 58=1*7+2*9+3*11

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

A (2*3)　　B (3*2)　　C (2*2)

$$z_1^{(2)} = w_{1,0}^{(1)} a_0^{(1)} + w_{1,1}^{(1)} a_1^{(1)} + w_{1,2}^{(1)} a_2^{(1)}$$

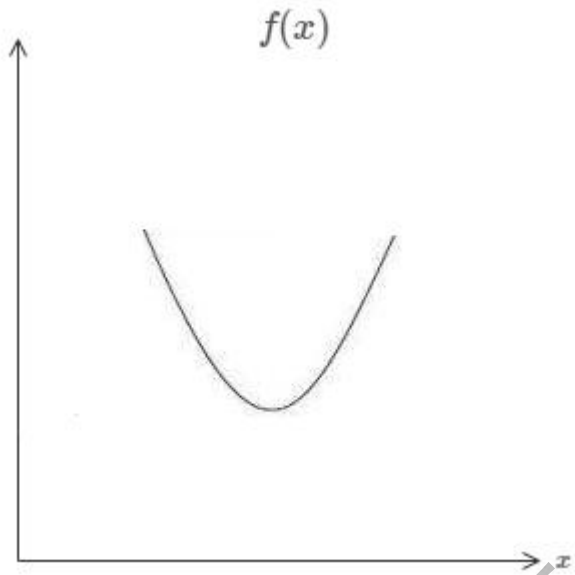
3.3：Universality



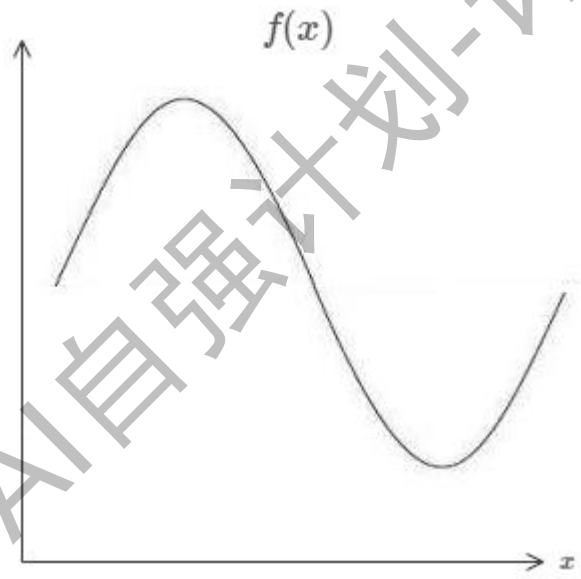
Universality → “全包了”性质

好霸气的词　　师兄的翻译

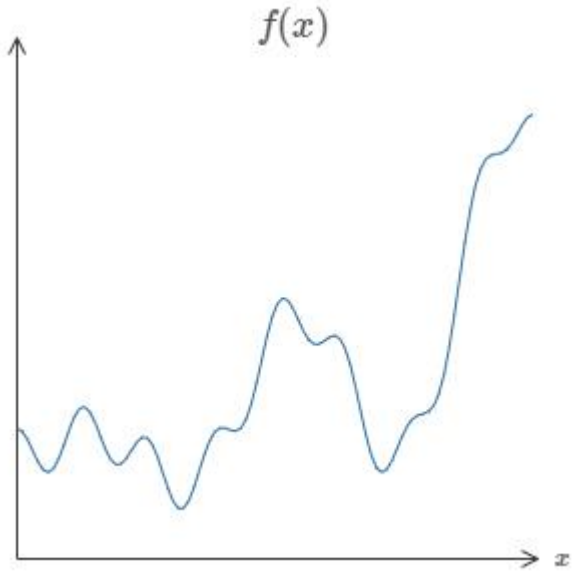
怎么讲？



二次函数？

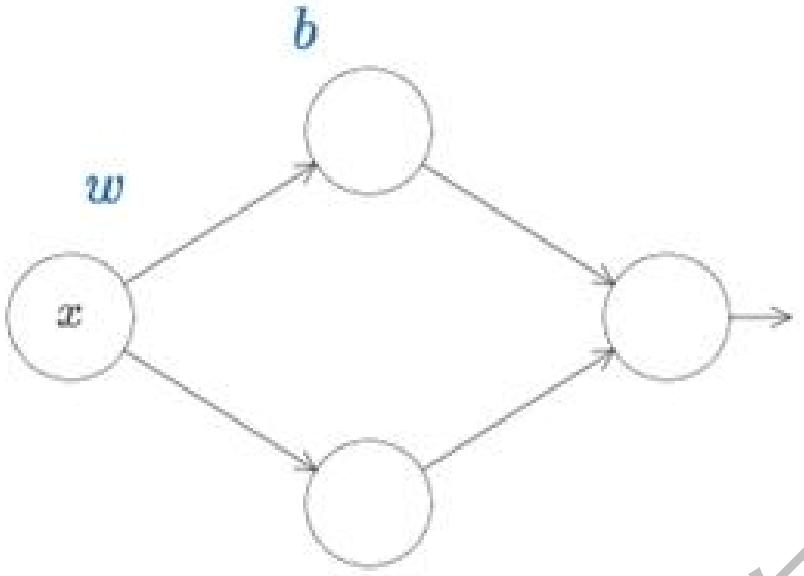


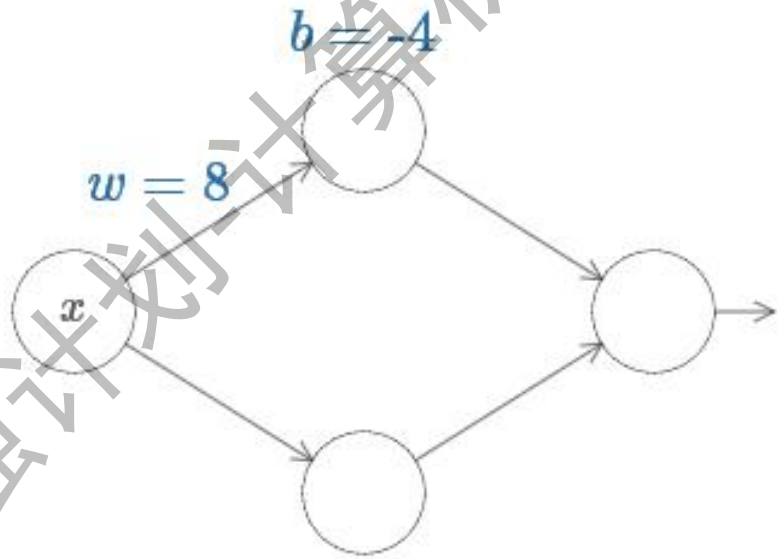
常数函数+sin？

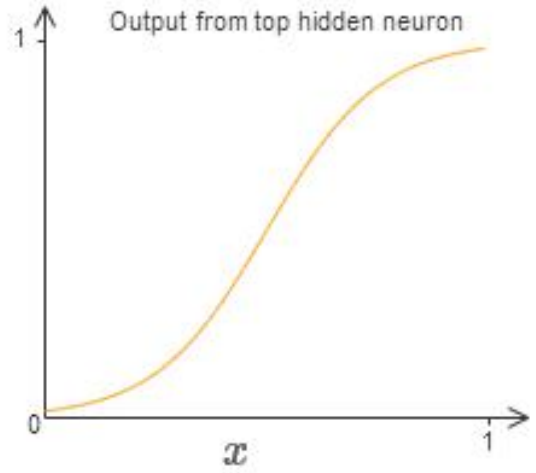


$$f(x) = 0.2 + 0.4x^2 + 0.3x\sin(15x) + 0.05\cos(50x)$$

3.3：Universality-证明1





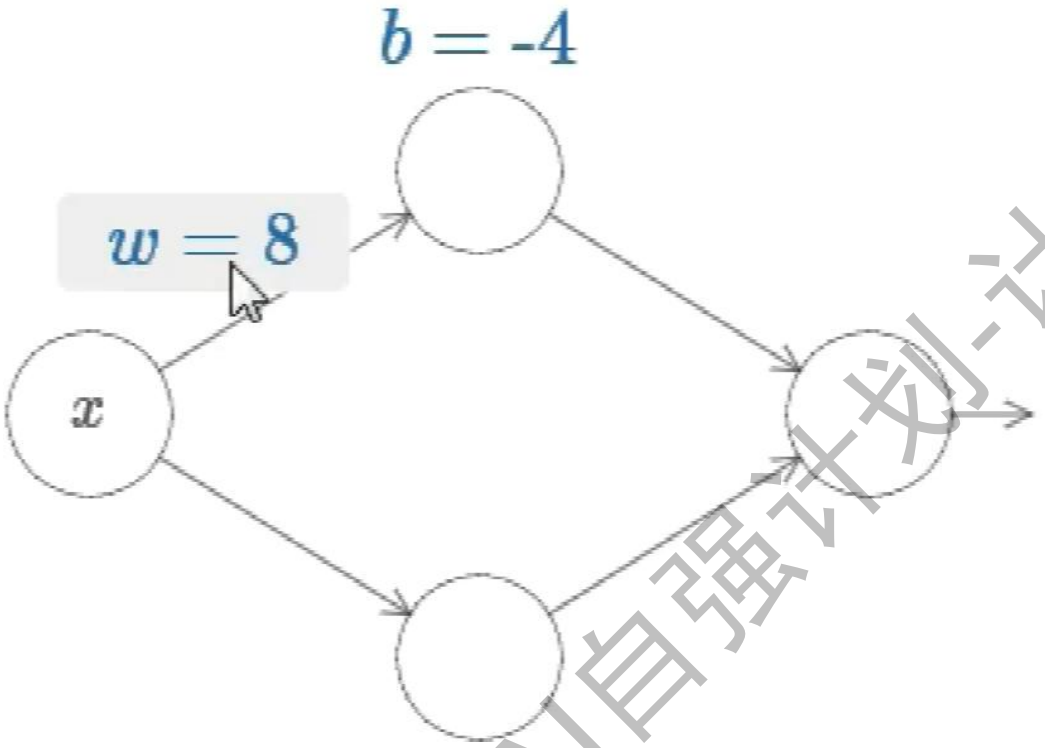


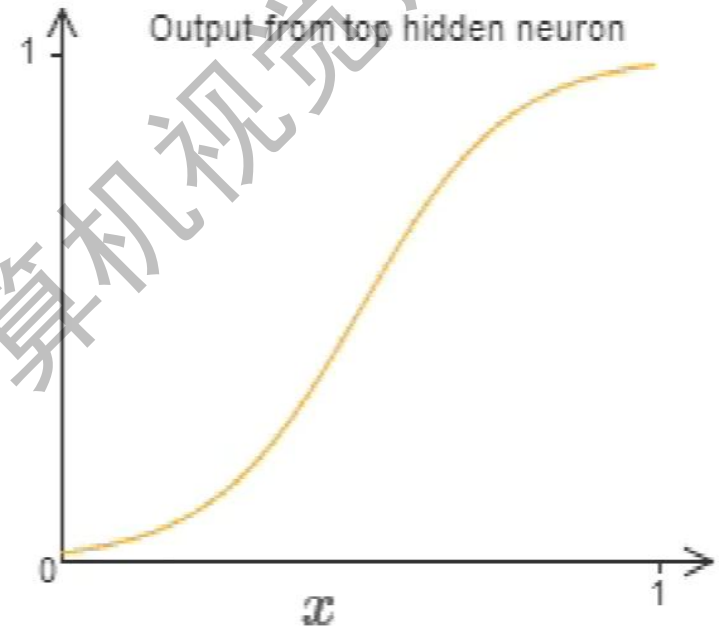
简化模型到只有2个参数

此时的函数表达式写出为：

$f(x) = g(8x - 4)$　　$g(\cdot)$代表sigmiod函数

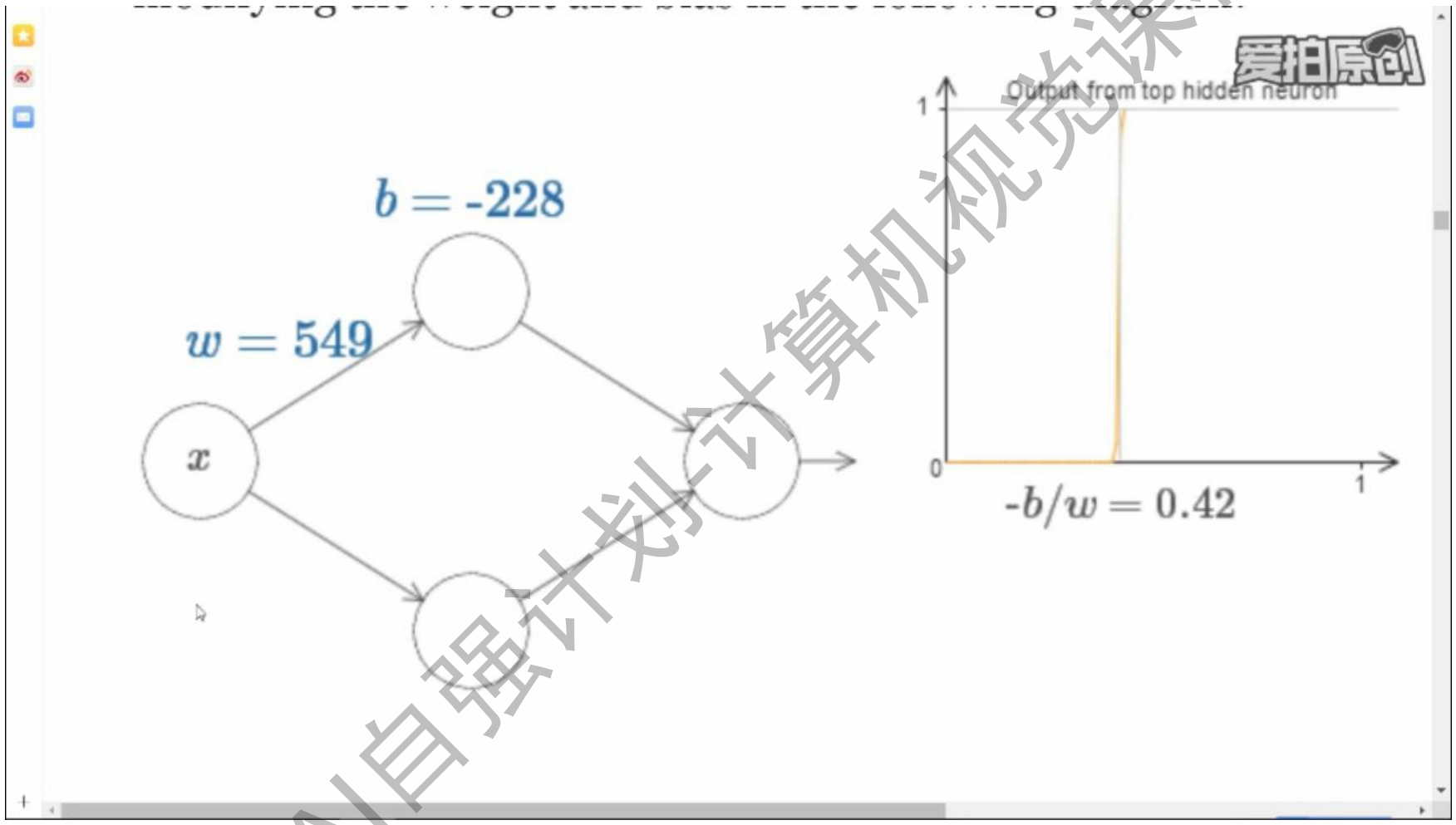
3.3：Universality-证明2





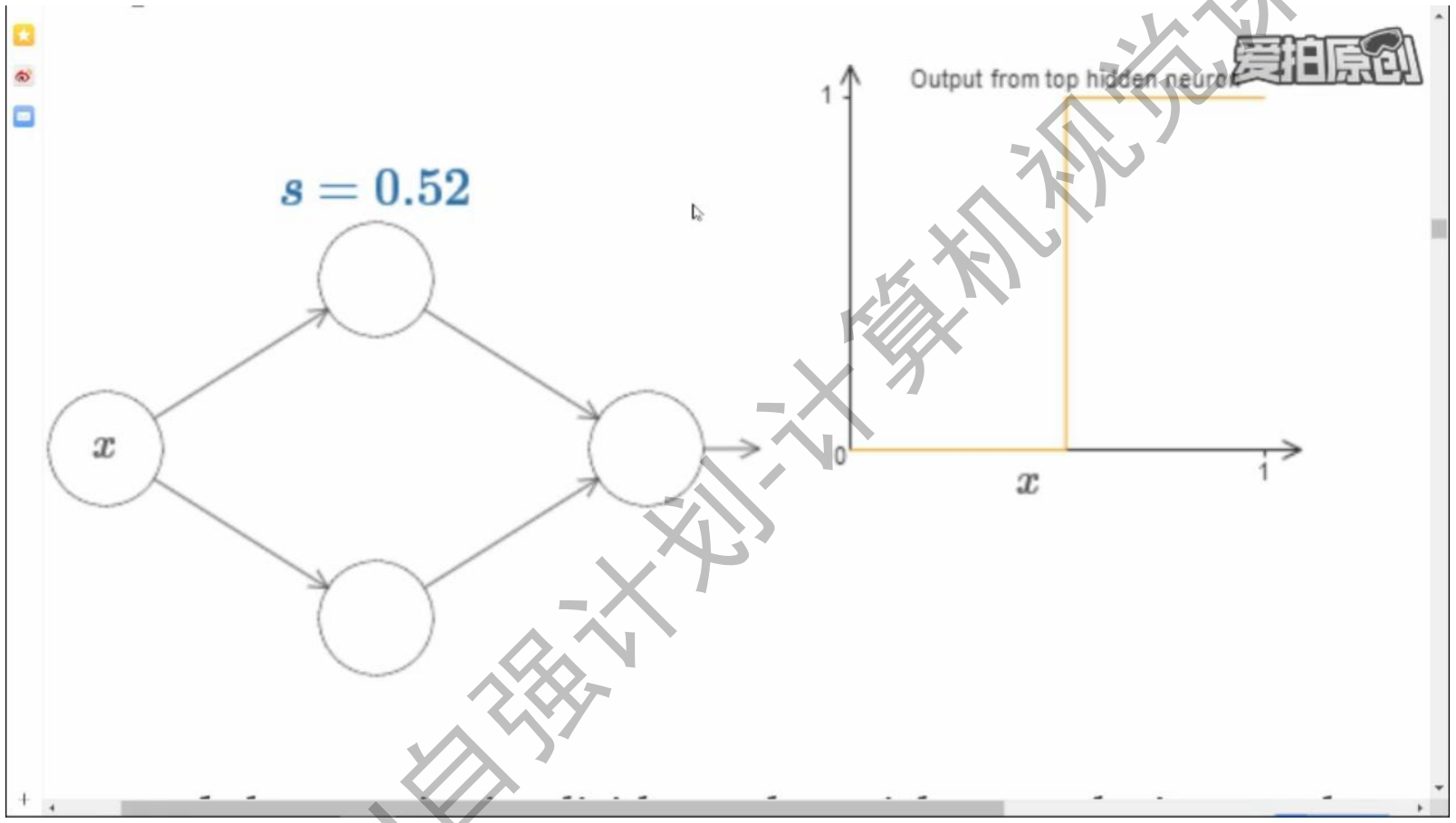
当w取值足够大
会出现理想的激活函数

3.3：Universality-证明3



激活函数的“跳变点”
出现在哪是有什么因素决定的？

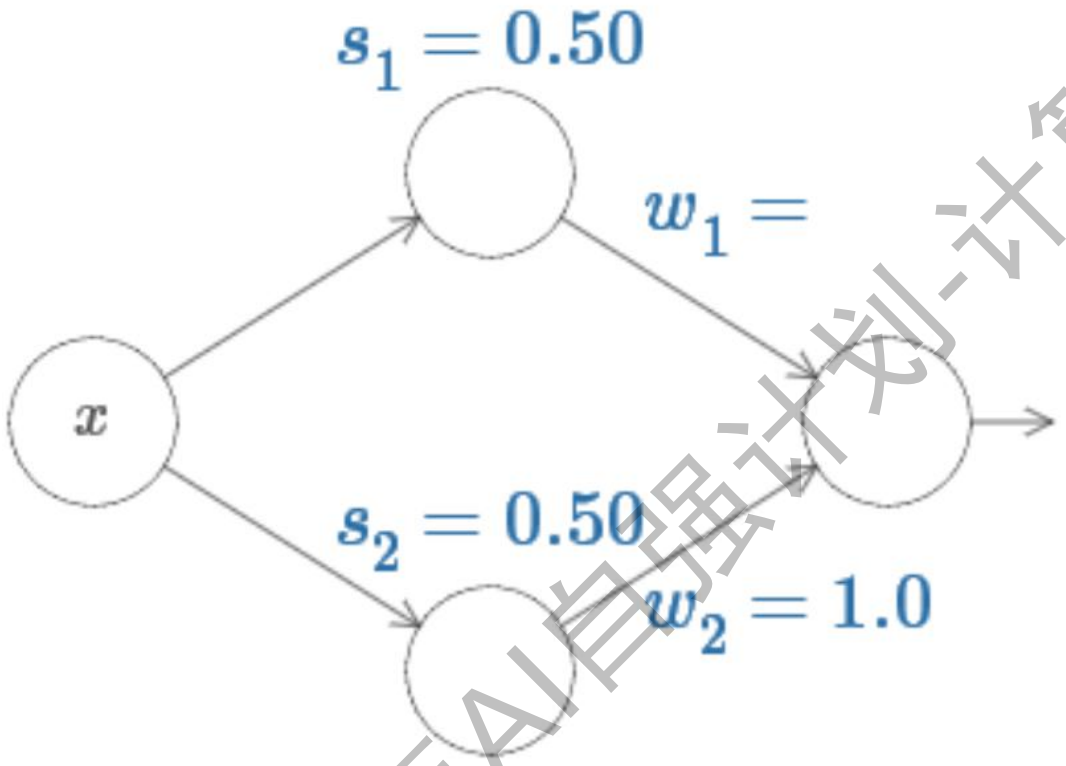
3.3：Universality-证明4



我们把跳变点设置为新的参数s（stride，步长）
并假设我们可以直接控制s来设定“跳变点”的准确位置？

3.3：Universality-证明5

为了画出更复杂一些的图像，我们把模型稍微扩充一些如下？



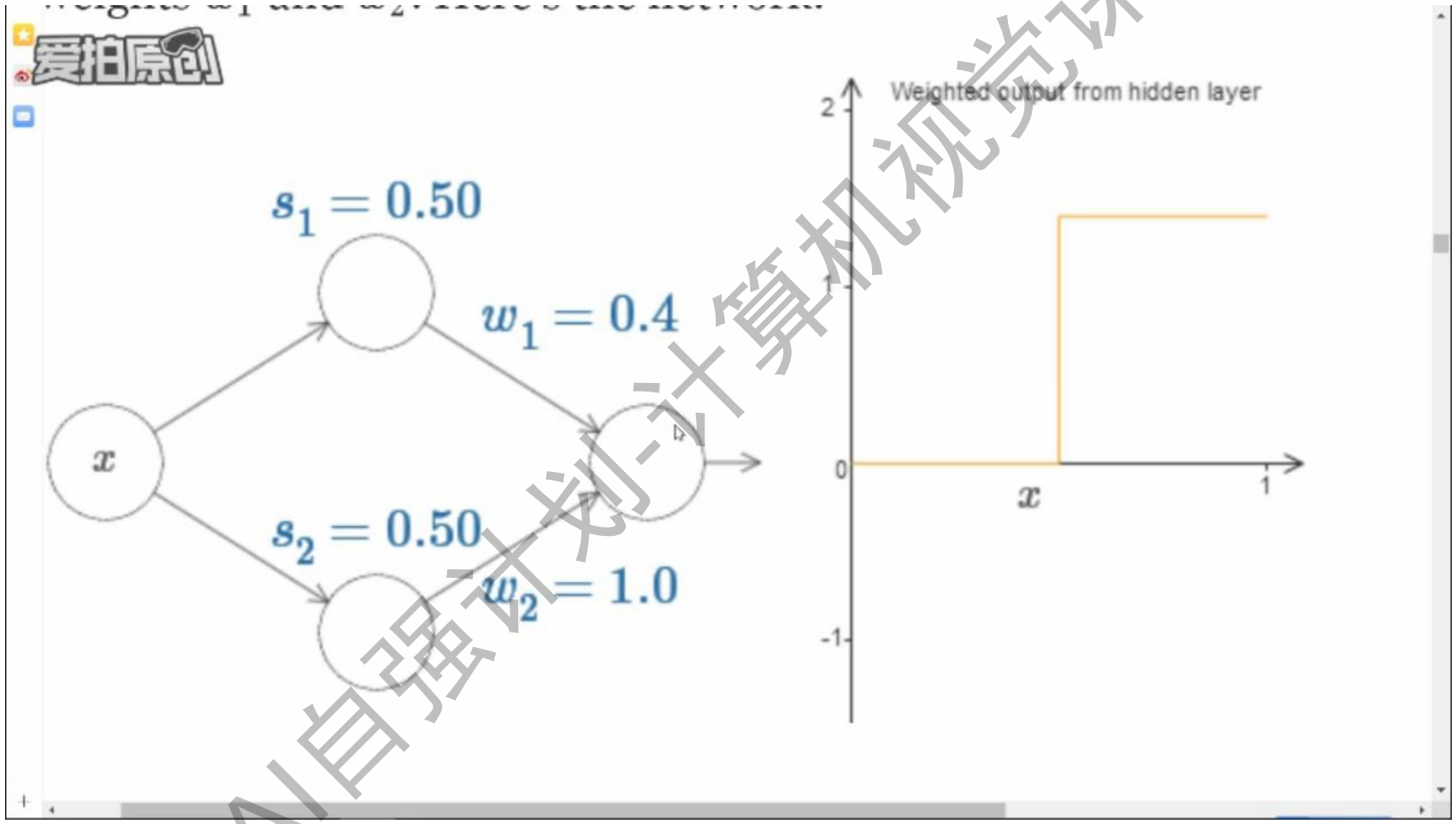
1、启用了全部神经元

2、隐藏层的2个神经元在与输出层相连时多出了两个权值w1和w2

3、假设输出层神经元没有激活函数

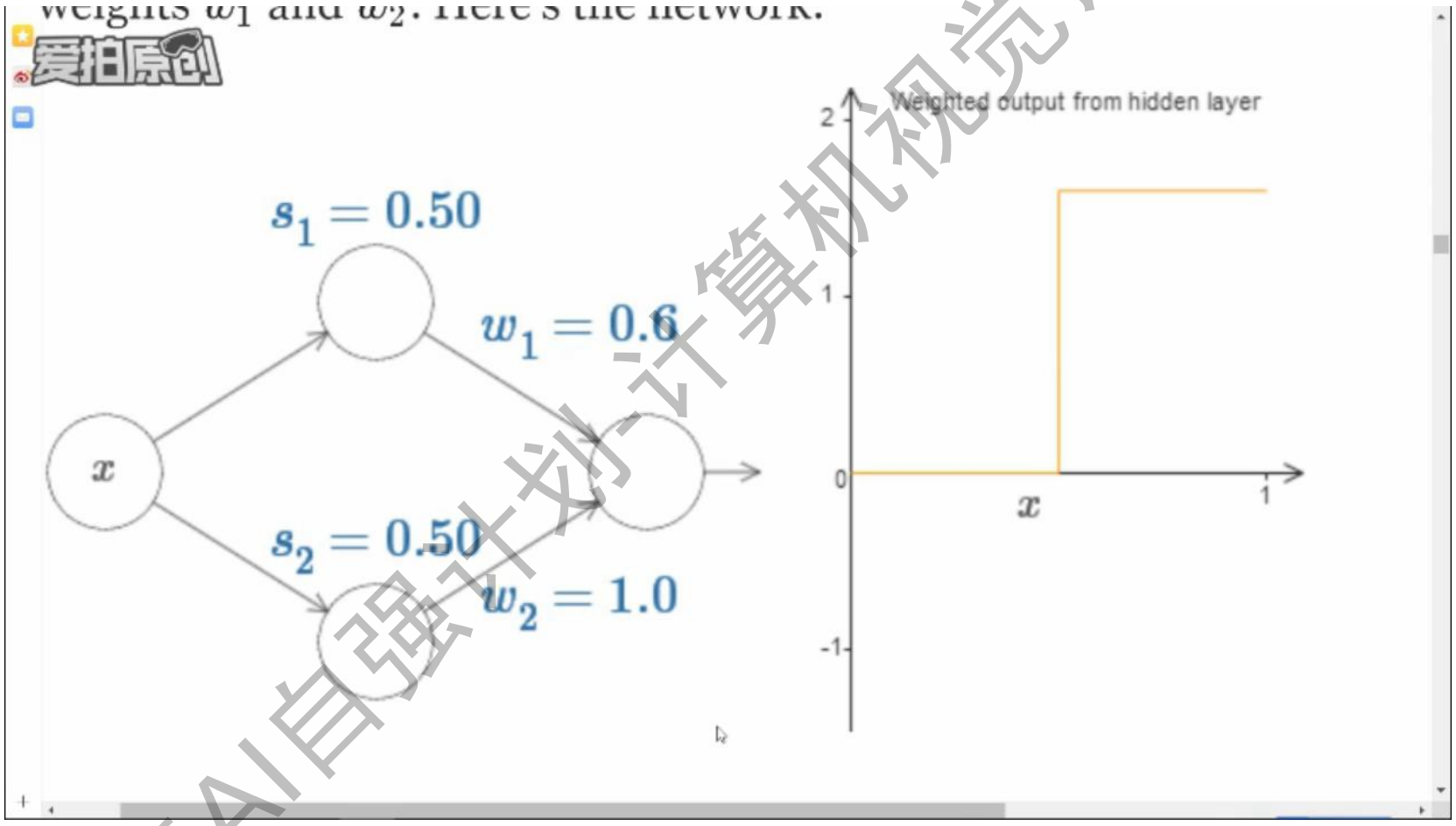
4、并且我们可以直接控制隐藏层的超参数s。

3.3：Universality-证明6



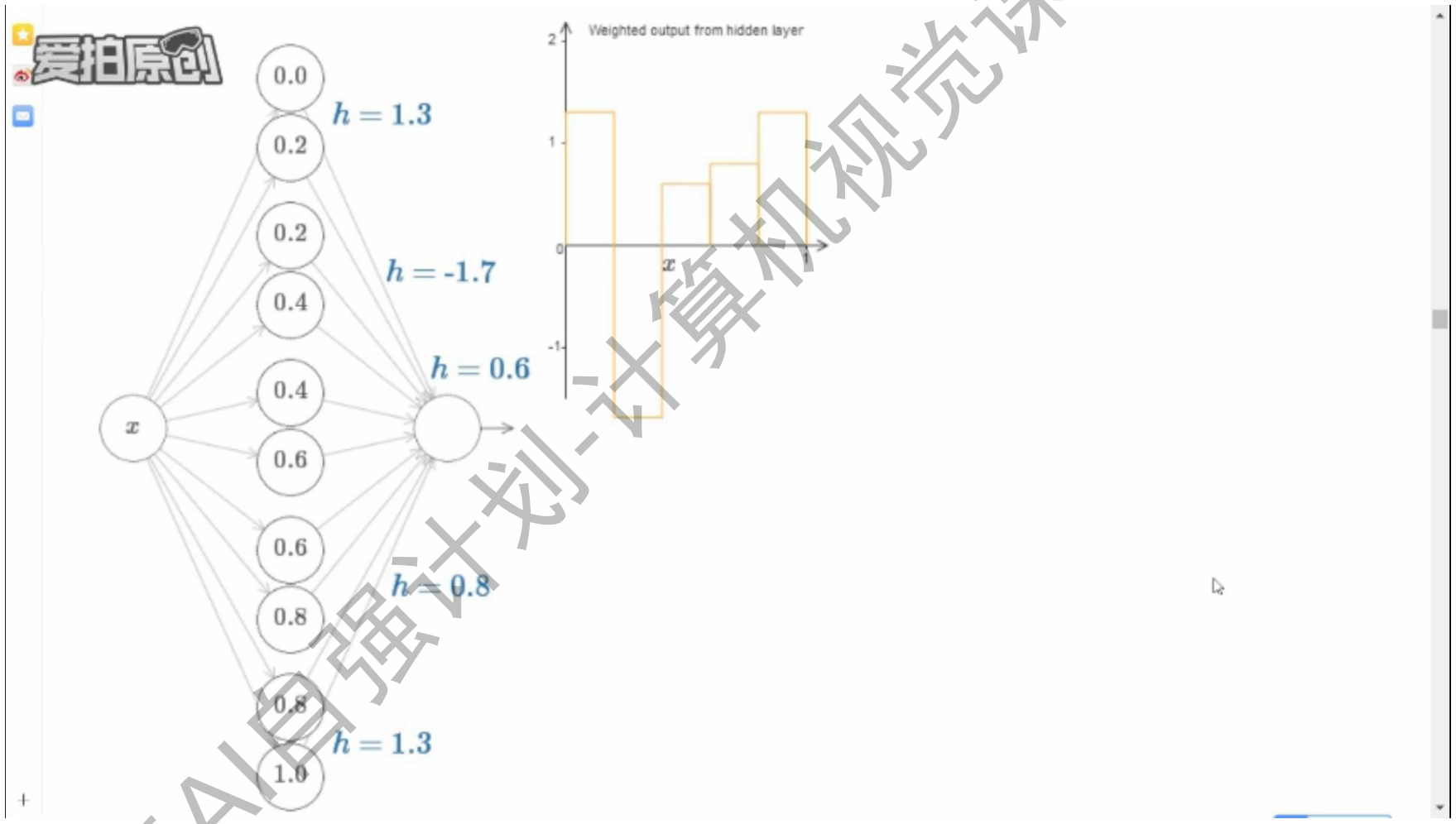
为了便于观察，我们先将s1、s2、w2的取值固定
仅改变w1，看看会发生什么

3.3：Universality-证明7



如果我们将w1和w2的取值设置为绝对值相等，但符号相反
同时将s1和s2设置为不同取值

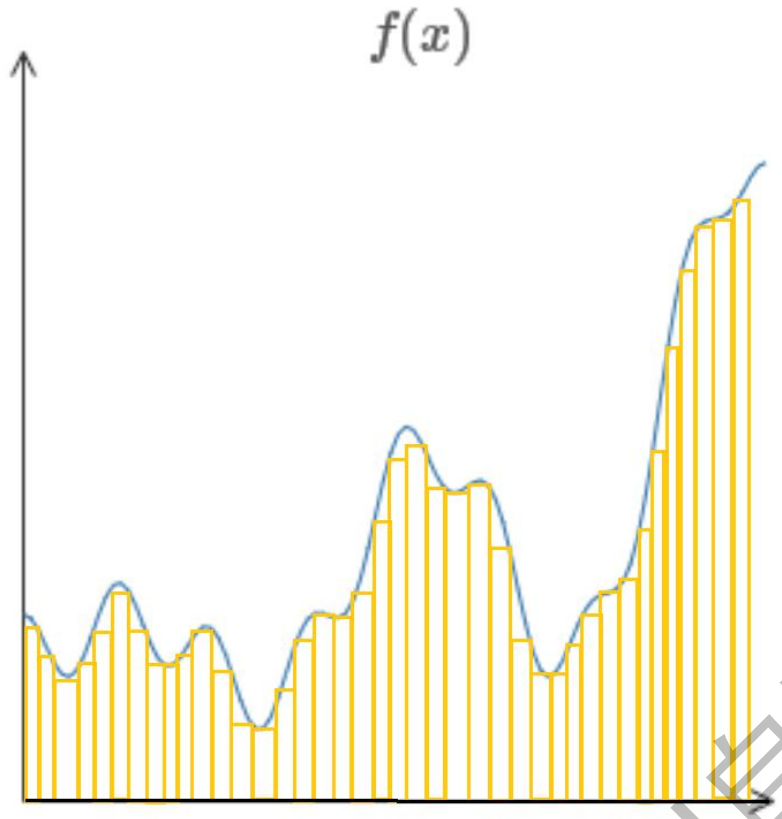
3.3：Universality-证明8



隐藏层每2个神经元可以画出一个形似方波信号的函数图像，
并且我们能够控制这个方波信号的宽度和高度。

3.3：Universality-证明9

http://neuralnetworksanddeeplearning.com/



沿着“极限”的思路往下走

只要我们无限的扩充隐藏层的神经元数量

我们就可以用无数的“小方波”

去无限的接近想要画出的任何函数图像。

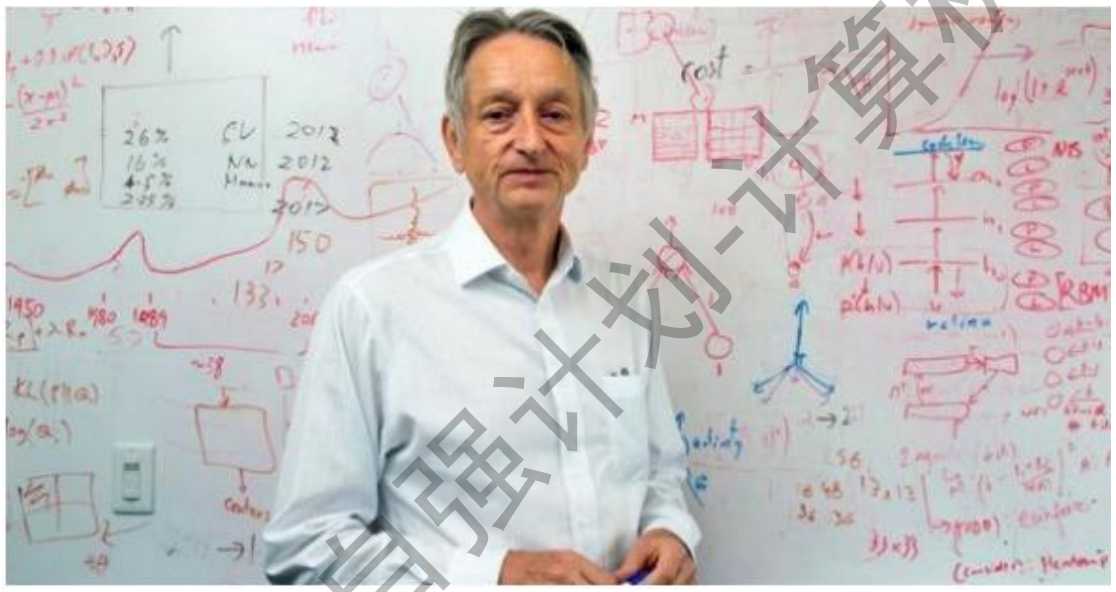
每一个“方波”相当于神经网络画出的“一笔”

理论上，我们可以画出无数笔，来一点点的描摹任何我们想要的决策边界。

4：AI大神-Geoffrey Hinton

1947

数学教师、昆虫学家、布尔代数



心理文学学位

AI博士

物理&化学

包工木匠

吵出自我　　物理&生理　　建筑学

4：AI大神-Geoffrey Hinton

第一次接触AI

谷底进入

引领高潮

不止如此：

Dynamic routing between capsules

S Sabour, N Frosst, GE Hinton - Advances in Neural Information ..., 2017 - papers.nips.cc

A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or object part. We use the length of the activity vector to represent the probability that the entity exists and its orientation to ...

☆ 99 被引用次数：260 相关文章 所有 10 个版本 »

5：反向传播（back propagation）
损失函数&one-hot

胖子的秤呢？

$$J(w) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_w(x^{(i)}) + (1-y^{(i)}) \log(1-h_w(x^{(i)}))]$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))]$$

二分类

多分类

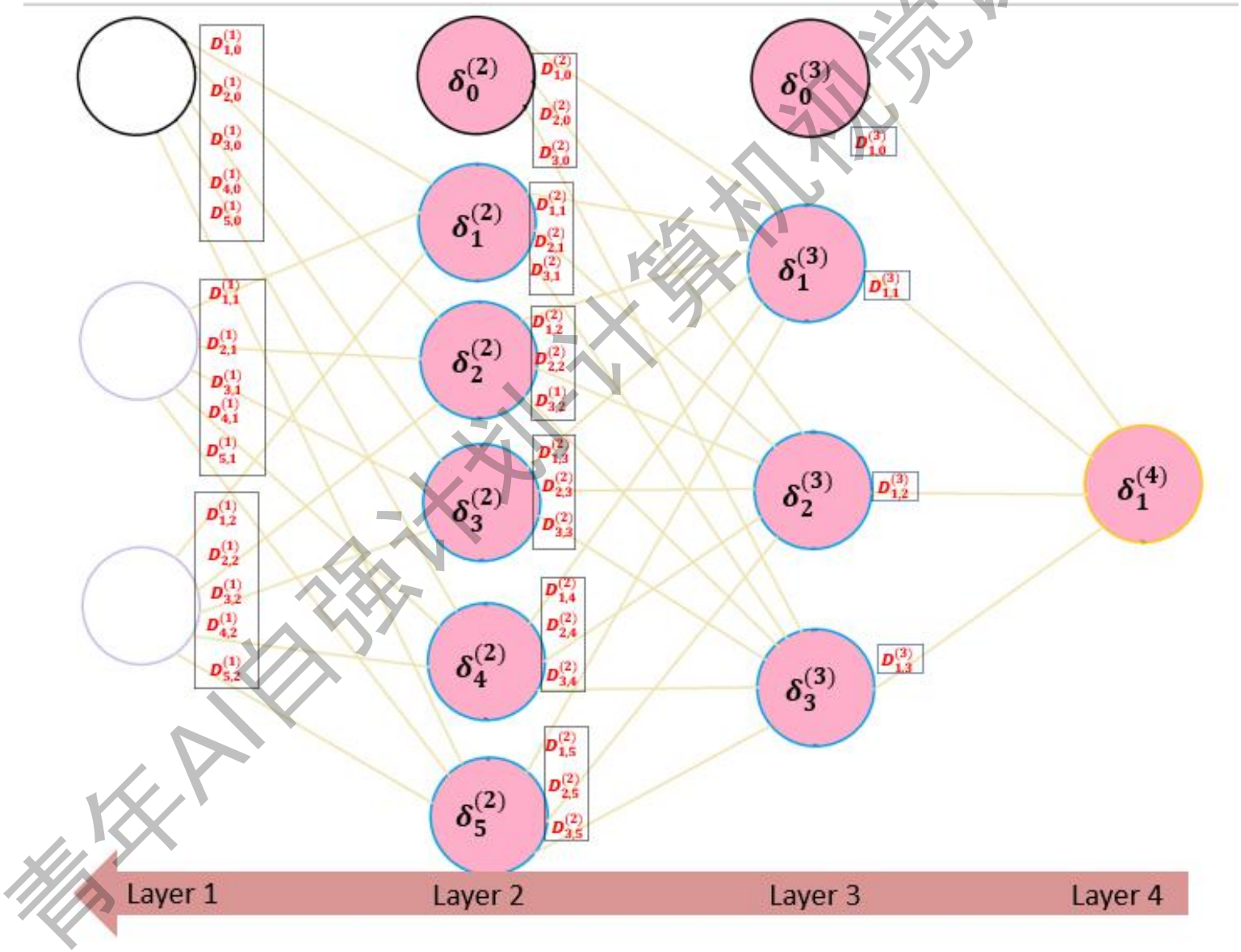
$y_{这个数为0} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$y^{(i)} = \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \cdots \\ y_K^{(i)} \end{bmatrix}$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$

5：反向传播（back propagation）

更新难题：
怎么求导



反向传播示意图

5.2：BP（back propagation）结构总述

结构及符号

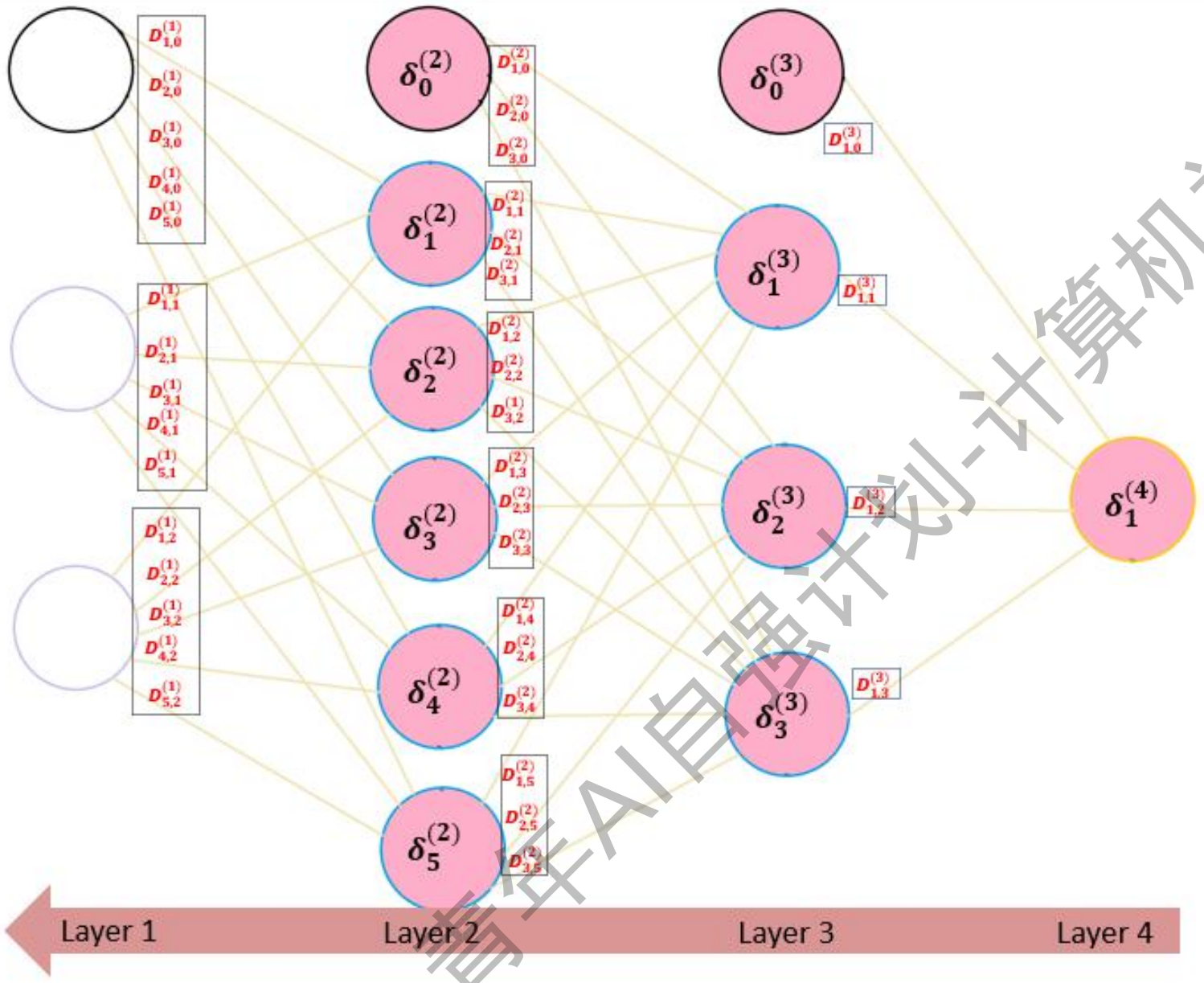
1. 在新的示意图中，原来的z、a、x、w虽然都没有写出来，但在运算中仍需要用到。本图新出现两个字母符号，分别是δ和D；
2. 对于模型中每一个神经元$a_j^{(l)}$（包括bias），都有一个$\delta_j^{(l)}$与之对应，表示该神经元的“误差”（error）；
3. 对于模型中的每一个参数$w_{ij}^{(l)}$，都有一个相应的$D_{ij}^{(l)}$与之对应，表示$w_{ij}^{(l)}$的导数（即梯度下降法中的$\frac{\partial}{\partial w_{ij}^{(l)}} J(w)$）；

5.2：BP-计算流程

流程：

1. 反向传播计算方向与前馈传播是相反的，即$\delta_j^{(l)}$是由其相邻深层网络中的神经元$\delta_j^{(l+1)}$计算得到的。如$\delta^{(3)}$为$\delta^{(4)}$运算后得到的；
2. 计算误差$\delta^{(l+1)}$的目的，是为了继续运算，得到相邻浅层网络模型参数“$w_{ij}^{(l)}$”的导数$D_{ij}^{(l)}$，并作为更新模型的依据。
3. 我们没必要计算$\delta^{(1)}$是因为在第1层（输入层）之前，再无需要更新的参数；
4. 反向传播能够计算的前提，是已经构造好了网络，并已经完成了前馈传播的计算。

5.2：BP-计算公式



本例中

$$\delta^{(4)} = a^{(4)} - y$$

$$\delta^{(3)} = (w^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$$

其中：$g'(z^{(3)}) = a^{(3)} .* (1 - a^{(3)})$

$$\delta^{(2)} = (w^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$$

一般的

$$\delta^{(l)} = (w^{(l)})^T \delta^{(l+1)} .* g'(z^{(l)})$$

$$g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$$

5.2：BP-误差计算的三个疑问

1、为什么要转置乘以该层的参数；

在求第l层误差$\delta^{(l)}$时，用第$l+1$层的误差乘以第l层模型参数的转置。可以理解为误差沿着激活信号“来时的路”反向传播回去。对应的参数矩阵转置之后与$\delta^{(l+1)}$相乘可以保证得到正确尺寸的$\delta^{(l)}$。

2、.*这个运算符是什么意思；

上述公式中的.*指两个矩阵按位相乘，要求做运算的两个矩阵的尺寸完全相同，新生成的矩阵尺寸仍旧不变，新矩阵每一位元素由两个原矩阵相应位置（相同角标）的元素直接相乘得来。在numpy中对应的命令为np.multiply()。

3、$g'(z^{(l)})$不知如何理解。

我们在高中时学过复合函数，对其求导需要采用“链式法则”，可以简单的概括为逐层求导。如果我们想要更新第l层的参数，那么可以将$a^{(l)}$理解为“原函数，而从第l层至输出层的前馈传播过程，本质上是构造了一个$a^{(l)}$的一个复合函数。所以如果想更新第l层的模型参数需要反向“逐层求导”直至第l层。

5.2：BP-偏导计算过程

参考

求每个样本（共m个）上的导数并求和

目标：$D_{ij}^{(l)}$

1、第1个样本上计算：$\Delta_{ij}^{(l)} = a_j^{(l)} \delta_i^{(l+1)}$

2、计算剩余m-1个样本，并累加 $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

3、累加和过除以m $D_{ij}^{(l)} = \frac{1}{\mathrm{m}} \Delta_{ij}^{(l)}$

4、更新参数 $w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha D_{ij}^{(l)}$

5.2：BP-偏导计算过程

详解：

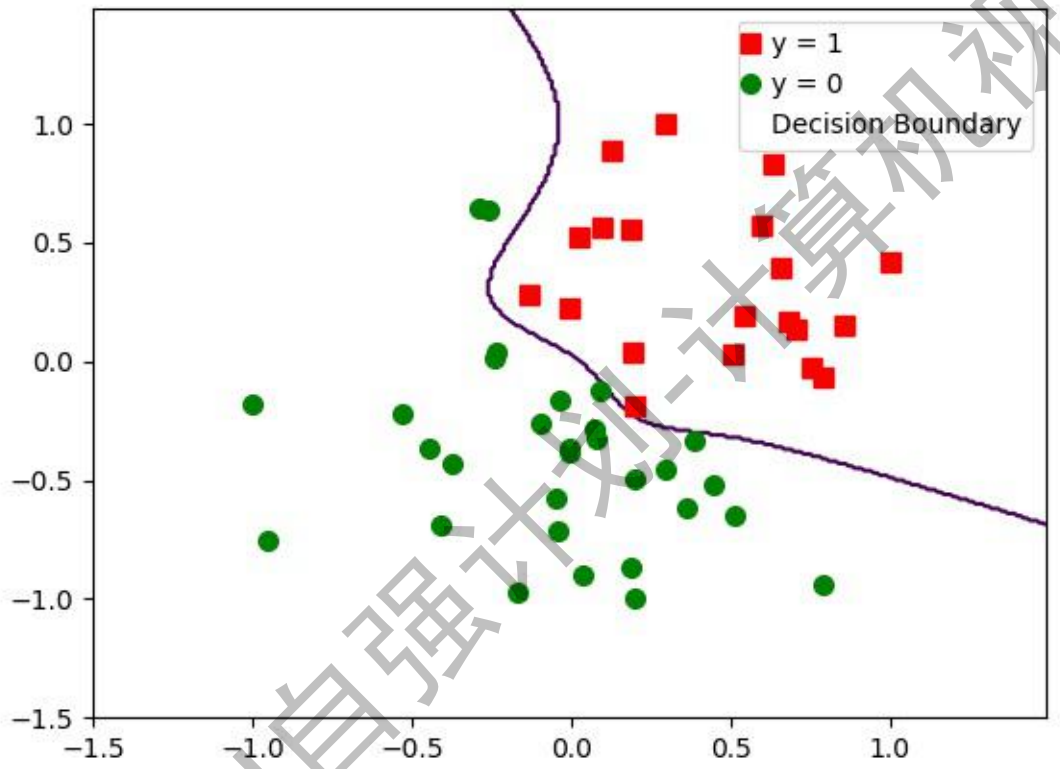
1. 假设共有m个样本，我们先在第1个样本上，计算出$w_{ij}^{(l)}$的偏导数暂记为$\Delta_{ij}^{(l)}$（δ的大写形式，读作delta），则$\Delta_{ij}^{(l)} = a_j^{(l)}\delta_i^{(l+1)}$；

2. 继续按照同样的办法逐次计算剩余m-1个样本上偏导数并累加到$\Delta_{ij}^{(l)}$中，即每做一次偏导，都有$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$；

3. 将累加了m次的$\Delta_{ij}^{(l)}$处以总样本数m，即可得到$D_{ij}^{(l)}$，即$D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)}$

然后，我们就可以按照上次课中同样的方法来更新参数的取值了，即：$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha D_{ij}^{(l)}$，其中$\alpha$为学习率。

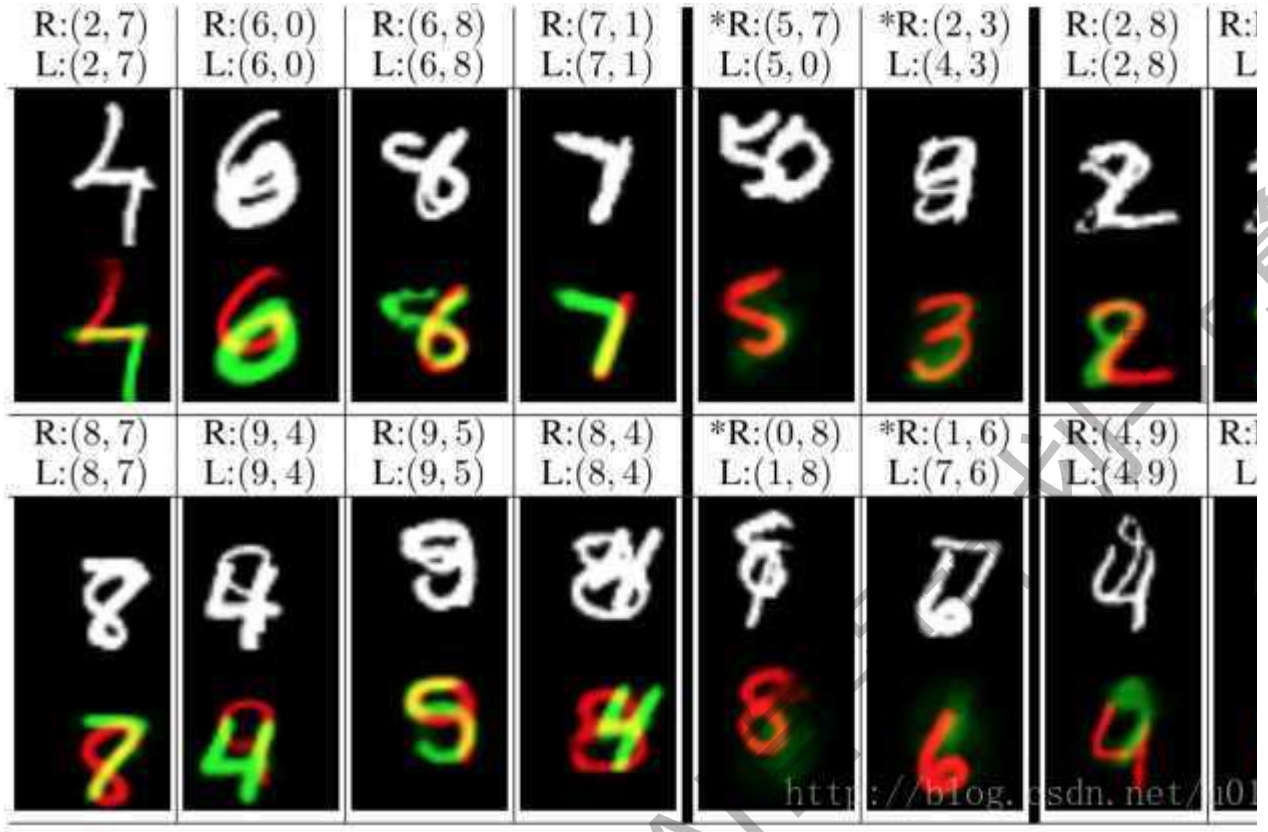
3.2：BP-训练结果



成功过拟合

6：延伸阅读-Capsule

6：延伸阅读-Capsule



虽然本次的 capsule 论文中并没有提到推翻反向传播的方法，但是 Hinton 已经在尝试找到新的方法来代替反向传播，并对当前的几种替代方案进行总结。2018.7.12 发表的文中对比了：

1 号选手，目标传播(Target-Propagation，TP)·

2 号选手，反馈对比(Feedback·Alignment，FA)·

3 号选手，目标差传播(Difference·Target·Propagation，DTP)·

4 号选手，误差反向传播(error·Back·Propagation，BP)

在各个公共数据集上的对比结果如下：

(a) MNIST

	FC		LC	
METHOD	TRAIN	TEST	TRAIN	TEST
DTP, PARALLEL	0.44	2.86	**0.00**	**1.52**
DTP, ALTERNATING	**0.00**	**1.83**	**0.00**	**1.46**
SDTP, PARALLEL	1.14	3.52	0.00	1.98
SDTP, ALTERNATING	0.00	2.28	0.00	1.90
AO-SDTP, PARALLEL	0.96	2.93	0.00	1.92
AO-SDTP, ALTERNATING	**0.00**	**1.86**	0.00	1.91
FA	**0.00**	**1.85**	**0.00**	**1.26**
DFA	0.85	2.75	0.23	2.05
BP	**0.00**	**1.48**	**0.00**	**1.17**
BP CONVNET	–	–	**0.00**	**1.01**

(b) CIFAR

	FC		LC	
TRAIN	TEST	TRAIN	TEST
59.45	59.14	28.69	39.47
30.41	**42.32**	28.54	39.47
51.48	55.32	43.00	46.63
48.65	54.27	40.40	45.66
4.28	47.11	32.67	40.05
0.00	45.40	34.11	40.21
25.62	**41.97**	17.46	37.44
33.35	47.80	32.74	44.41
28.97	**41.32**	**1.83**	**32.41**
–	–	**1.39**	**31.87**

ImageNet

METHOD	TOP-1	TOP-5
DTP, PARALLEL	98.34	94.56
SDTP, PARALLEL	99.28	97.15
FA	93.08	82.54
BACKPROPAGATION	**71.79**	**49.54**
BACKPROPAGATION, CONVNET	**63.93**	**49.17**

由图中可以看出：在 MNIST、CIFAR、ImageNet 上的测试结果都是 BP 最好。

论文：Assessing· the· Scalability· of· Biologically-Motivated· Deep· Learning· Algorithms· and· Architectures

链接：https://arxiv.org/pdf/1807.04587.pdf

7：NN vs MNIST

第三个小任务，棋逢对手：NN VS MNIST

```
x = tf.placeholder(tf.float32, ??)  #申请占位符 输入图像 N*784的矩阵
y_ = tf.placeholder(tf.float32, ??)  #申请占位符 输入label N*10的矩阵
```

```
A1 = tf.layers.dense(inputs=x,units=??,activation=tf.nn.??)
A2 = tf.layers.dense(inputs=A1,units=??,activation=tf.nn.??)
A3 = tf.layers.dense(inputs=A2,units=10,activation=tf.nn.?)
```

```
cross_entropy = -tf.reduce_sum(??)
```

```
train_step = tf.train.GradientDescentOptimizer(??).minimize(cross_entropy)
```

8.1：尾声-要坚持

让苍天知道我认输

主创团队的心路历程

200h、400h

为什么坚持

比比看谁被恶心的多一点

8.2：尾声-作业说明

作业1：用线性模型拟合MNIST数据集；
了解不同模型拟合能力的天然差别之外
熟悉一下TF框架的基本编程思路及用法。

作业2：BP神经网络拟合上一堂课例子（“小哥哥是否受欢迎”）的数据
用numpy一点点的写出前馈传播以及反向传播的整个过程，让大家能够熟悉课上所讲的公式。

作业3：BP神经网络拟合MNIST数据集
更接近真实情况，让大家能从细节中解放出来，
更宏观的认识算法

扫码加好友进群

关注直播间公告