深度前馈神经网络

试试理解计算机语言

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

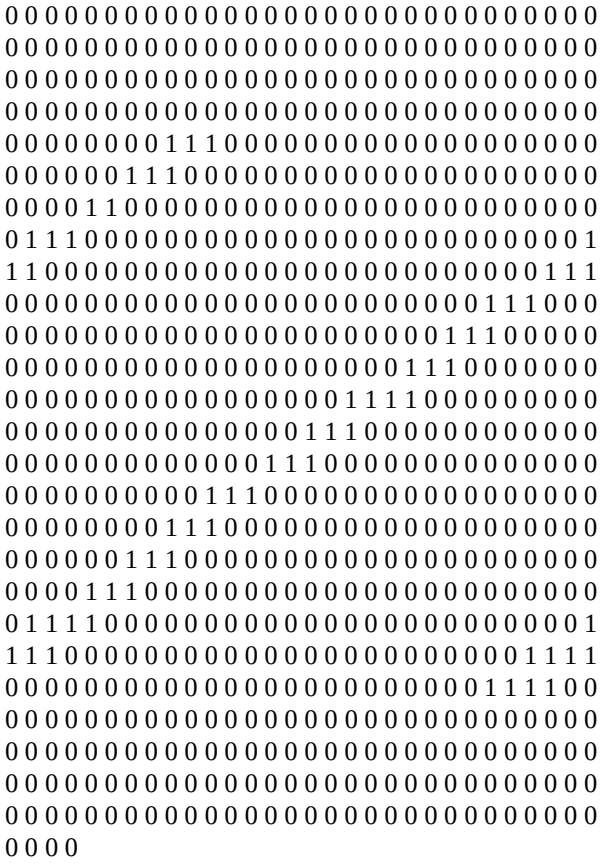
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

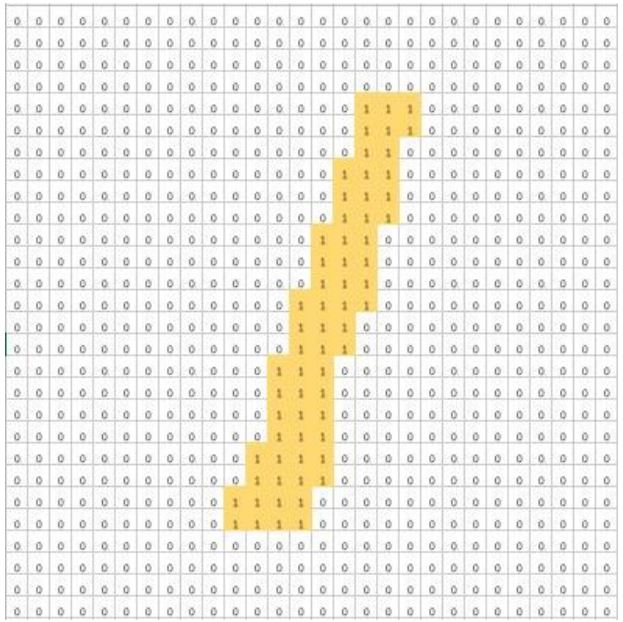
0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

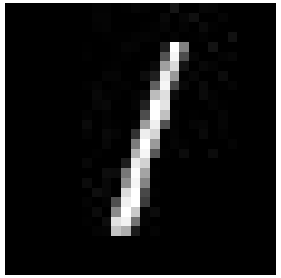
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

计算机眼里的图片

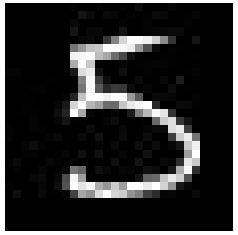


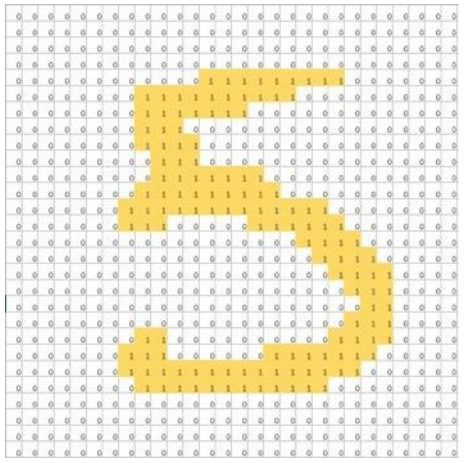


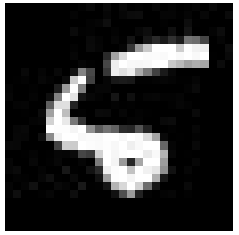


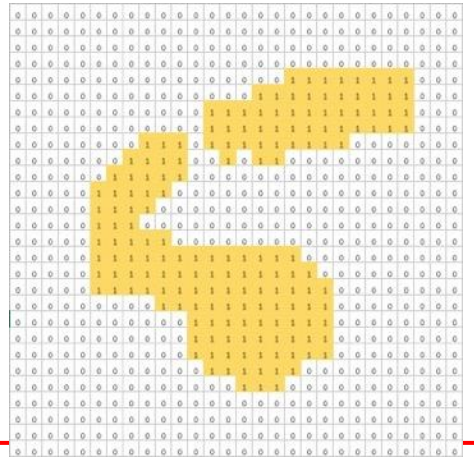
手写阿拉伯数字1

5 还是 6?

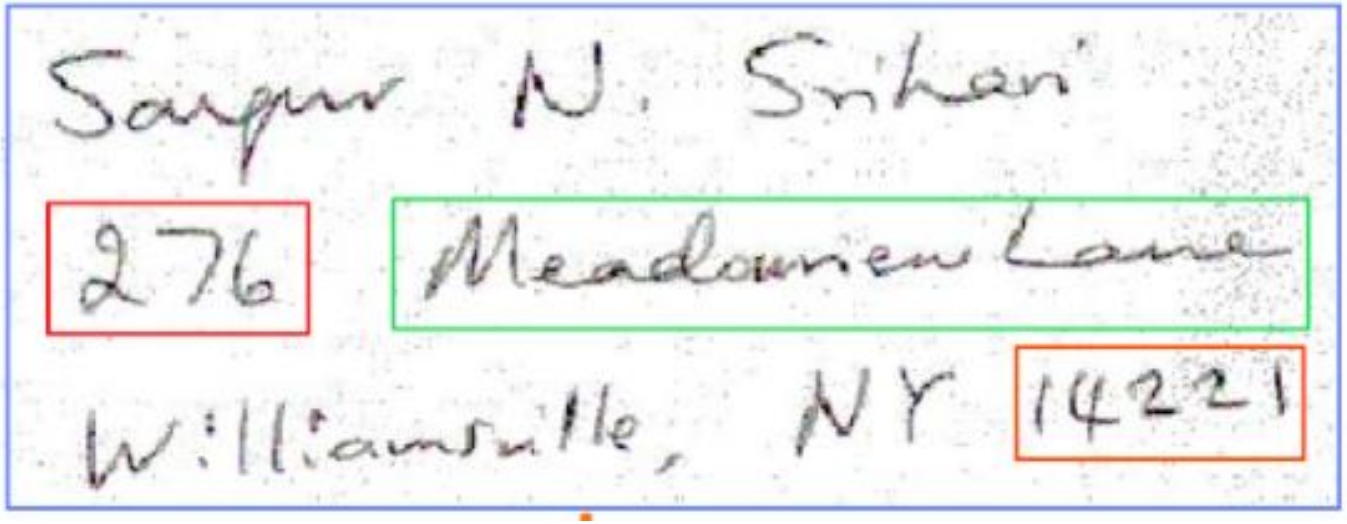








- 识别规则
  - 形态
  - 比划顺序

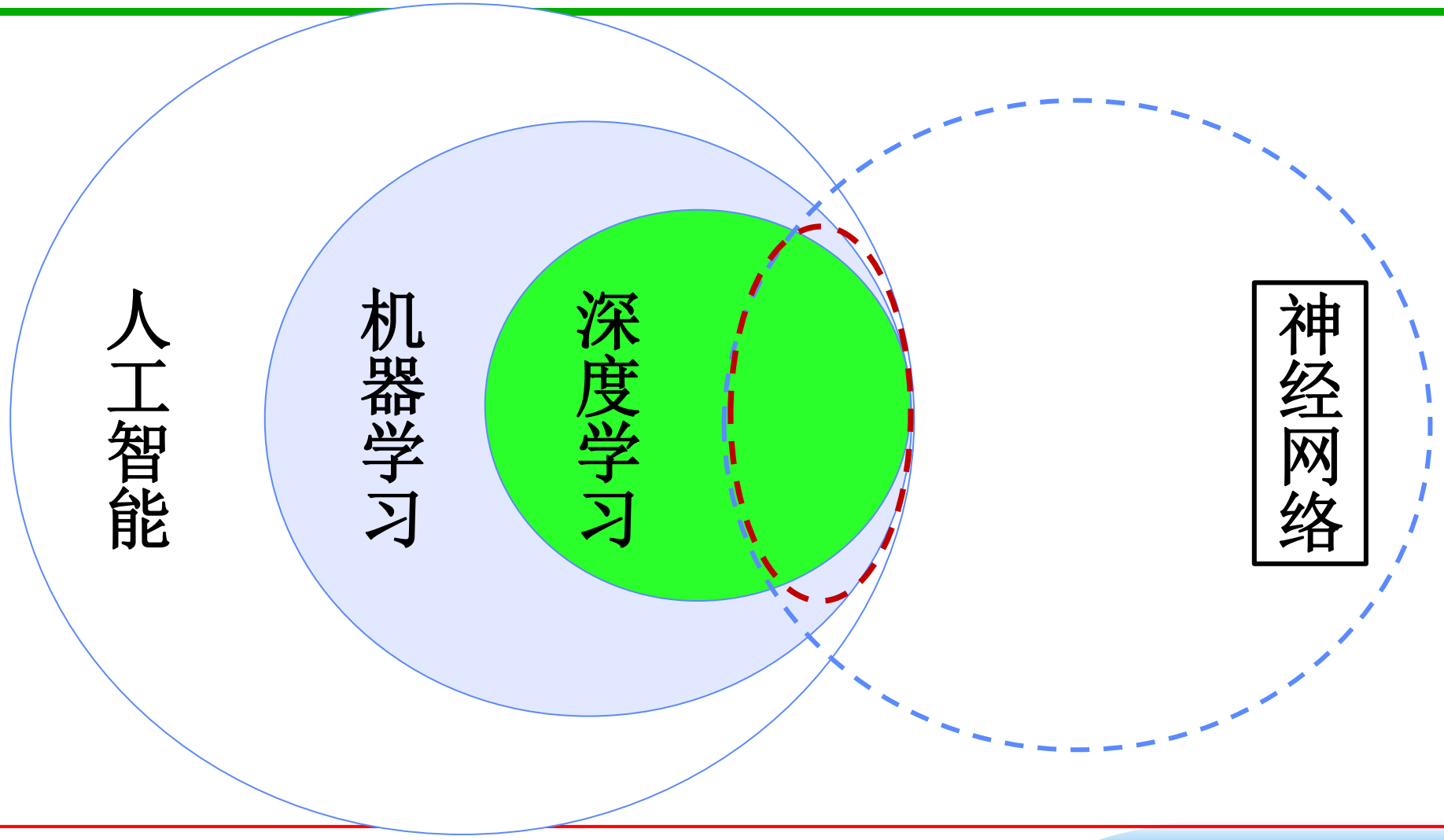


- 如何定义明确的识别规则?

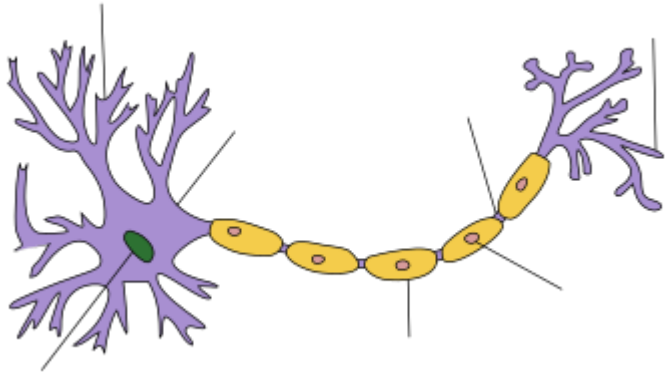
目录

神经网络的由来及发展

神经网络范畴

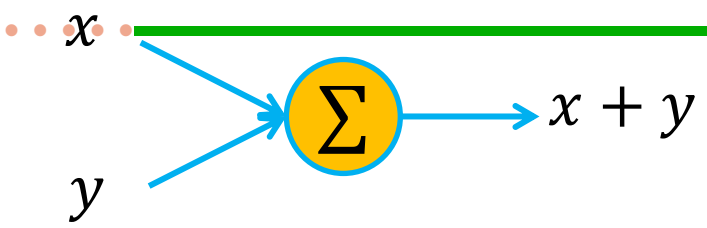


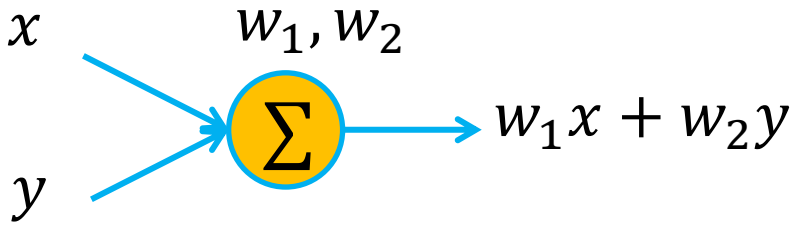
神经元 vs. 人工神经元



神经元　　人工神经元

线性运算

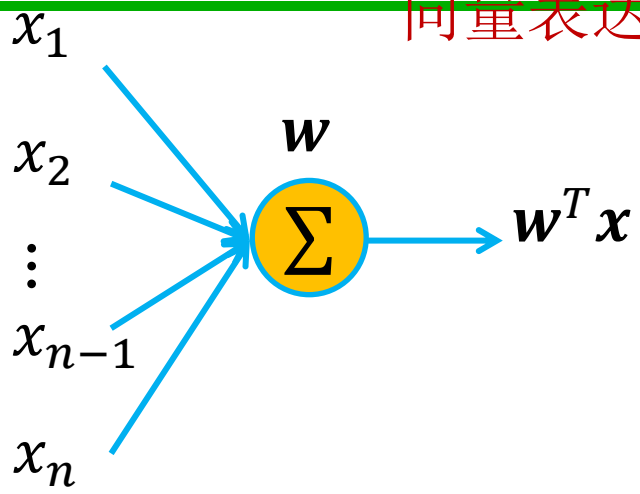


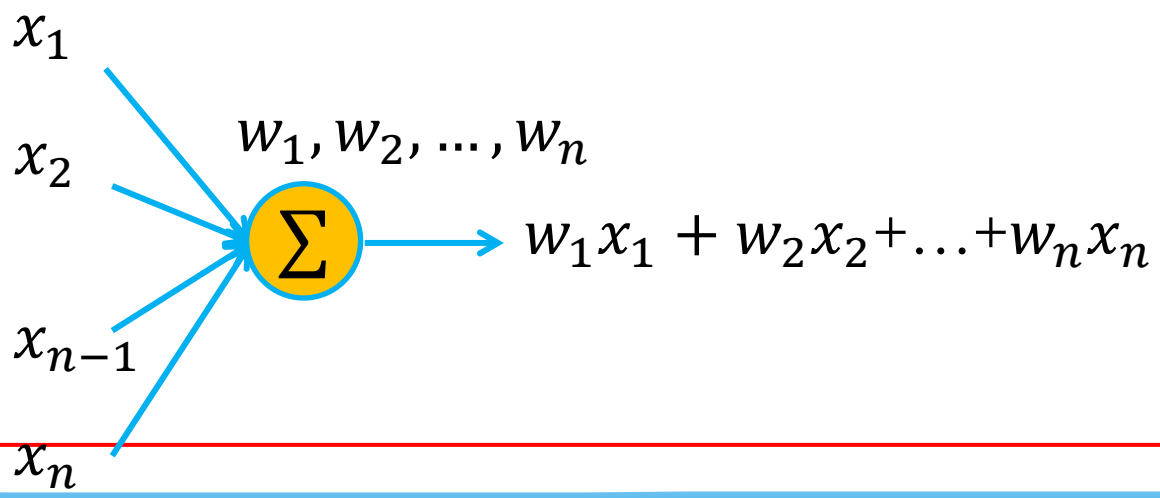


$$\boldsymbol{w}=[w_1, w_2, \ldots, w_n]$$

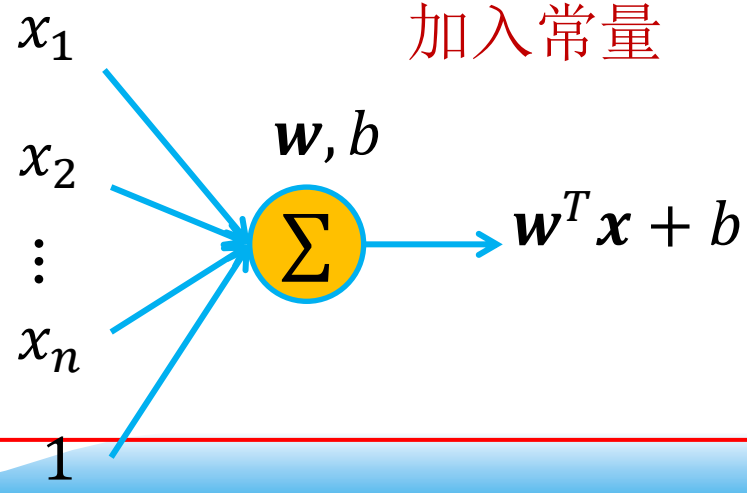
$$\boldsymbol{w}^T\boldsymbol{x} = \begin{matrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{matrix} [x_1 + x_2 + \cdots + x_n]$$

向量表达





加入常量



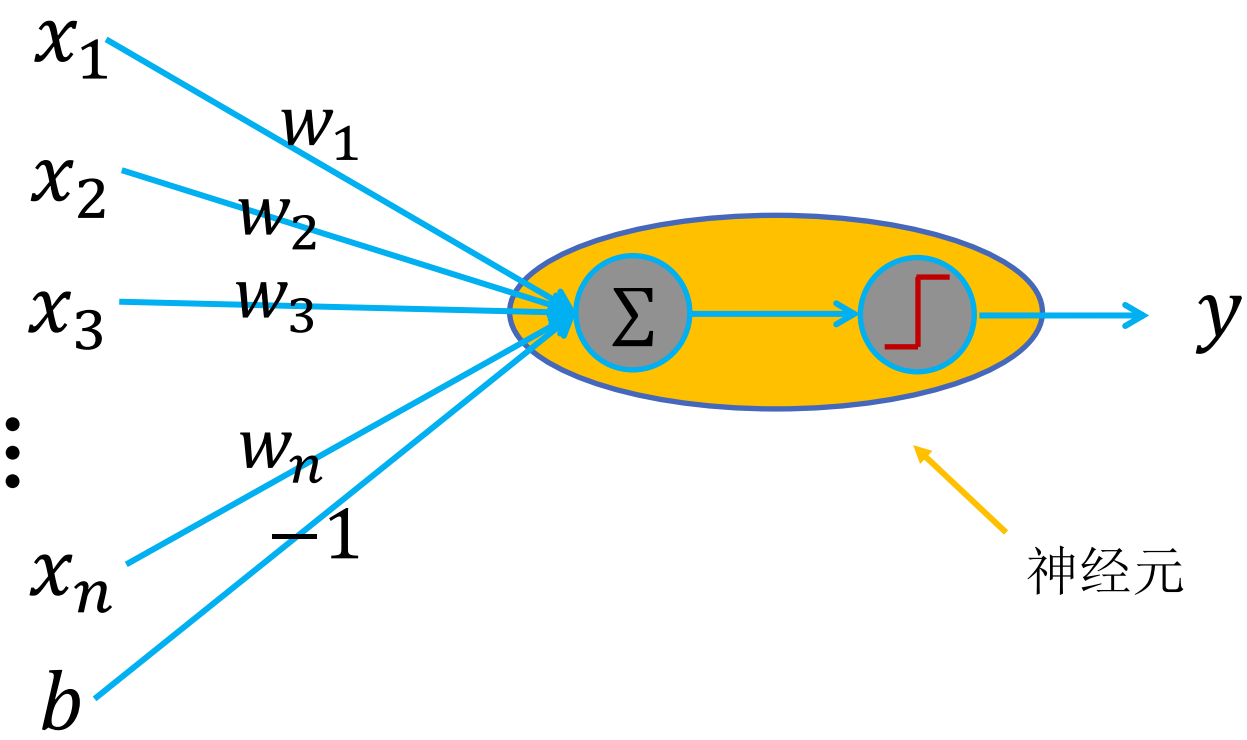
线性函数加入激活函数

输入　　权重　　求和　　激活函数　　输出

阶跃函数实现激活功能　$y=\begin{cases}1\ (x\geq 0)\\0\ (x<0)\end{cases}$

10
5
-1
-8

1
1
0
0

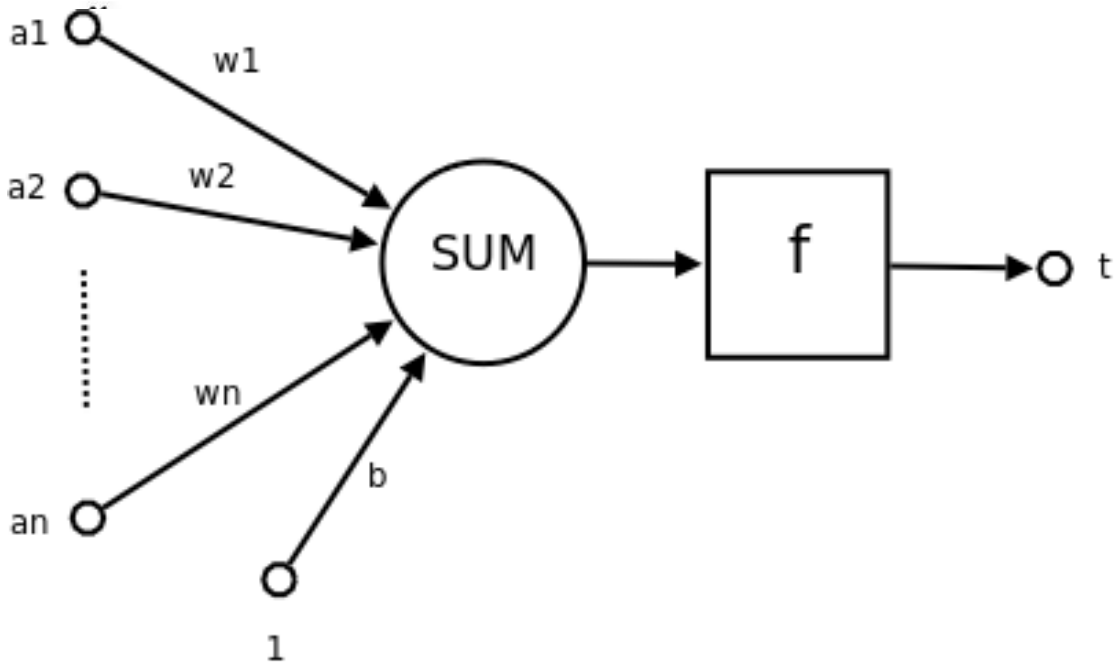


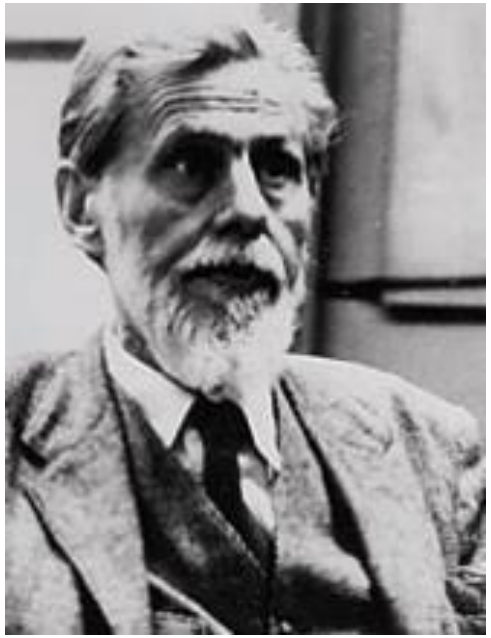
$$y=f(\sum_{i=0}^{n}w_ix_i-b)$$

$$=\begin{cases}\mathbf{0}\ if\ \sum_{i=0}^{n}w_ix_i<b\\\mathbf{1}\ if\ \sum_{i=0}^{n}w_ix_i\geq b\end{cases}$$

$$w_1x_1+w_2x_2+w_3x_3+\cdots+w_nx_n-b \Rightarrow \sum_{i=0}^{n}w_ix_i-b$$

1943 年，人工神经元模型MP被提出





沃伦.麦卡洛克

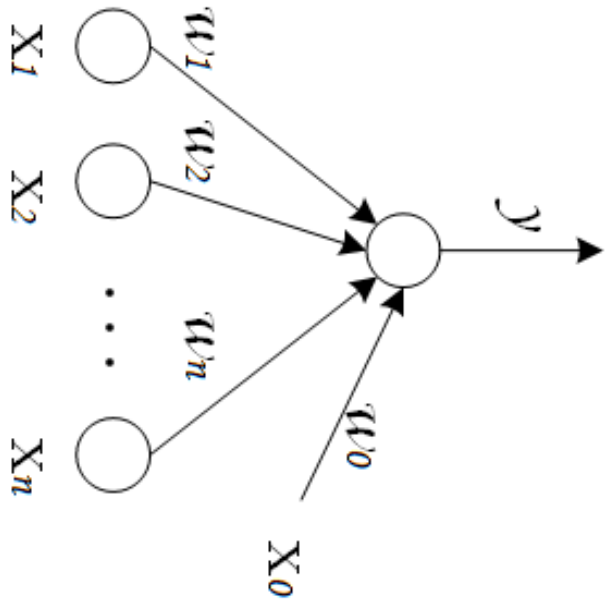
沃尔特.皮茨

$$o=\begin{cases}1 & \sum_{j=0}^{n} w_j x_j - b \geq 0 \\ 0 & \sum_{j=0}^{n} w_j x_j - b < 0\end{cases}$$

- 多输入单输出
- 空间整合
- 兴奋性和抑制性
- 阈值特性

1958年，Rosenblatt提出感知器

- 网络结构
  - 由两层神经网络组成
  - 输入层接收外界输入信号
  - 输出层是MP神经元
- 引发了神经网络第一次高潮

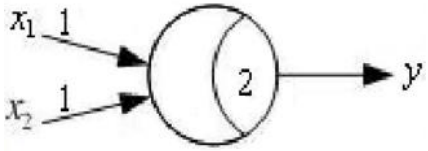


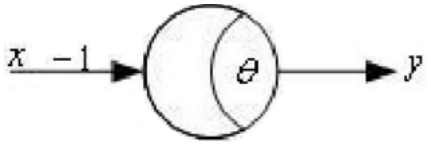
$$y = \mathrm{sgn}(\boldsymbol{w} \cdot \boldsymbol{x})$$

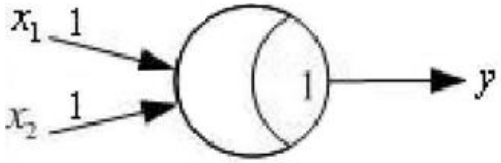
$$\mathrm{sgn}(y) = \begin{cases} 1, & \text{if} \quad y > 0 \\ 0, & \text{otherwise} \end{cases}$$

1969年，神经网络开始步入冰河期

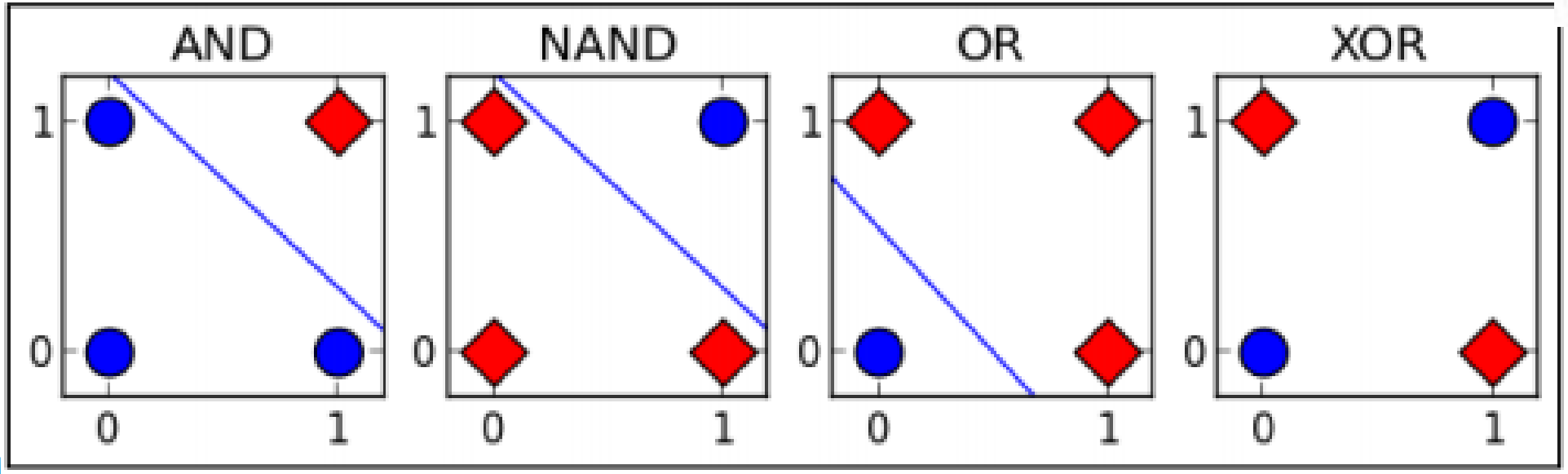
- Minsky出版《Perceptron》
  - 单层神经网络只能解决线性可分问题
  - 单层神经网络无法解决XOR问题







?



神奇的异或运算

Magic xor Box

小考题 1：出现奇数次的数字

有一堆数字，

20243596817437850l9

哪个数字只出现了一次？

	数字	1	2	3
数字	二进制	01	10	11
1	01	00	11	10
2	10	00	00	01
3	11	10	01	00

对于数字组成的任意序列，如果只有一个数字出现了基数次，如何找到这个数字？

对于 221　　2 **xor** 2 **xor** 1 = 1

对于 22113　　2 **xor** 2 **xor** 1 **xor** 1 **xor** 3 = 3

小考题 2：交换数值

a = 2
b = 3

列出计算公式
只对 a 和 b 进行运算，使

a = 3
b = 2

a 和 b 的值并不提前给出，如何计算?

a = a **xor** b
b = a **xor** b
a = a **xor** b

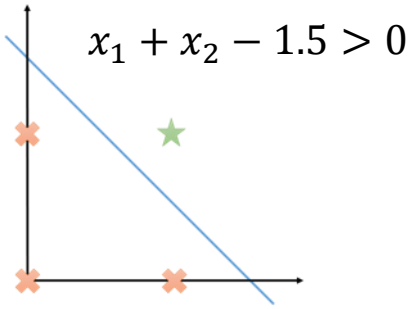
a = 2 **xor** 3 = 1
b = 1 **xor** 3 = 2
a = 1 **xor** 2 = 3

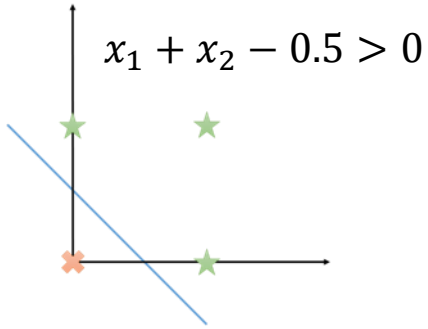
线性可分与不可分

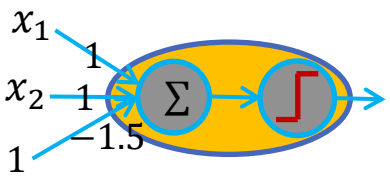
线性可分

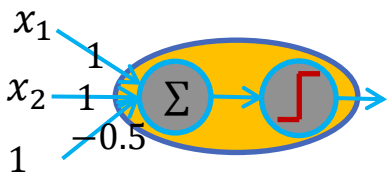
x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1



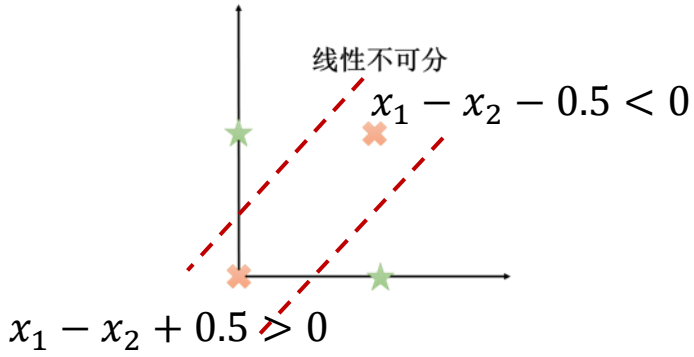


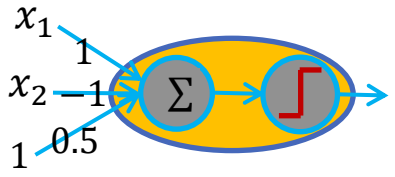


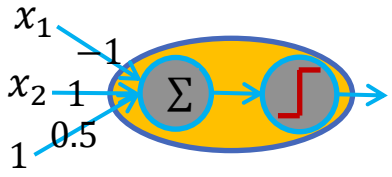


线性不可分

x_1	x_2	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

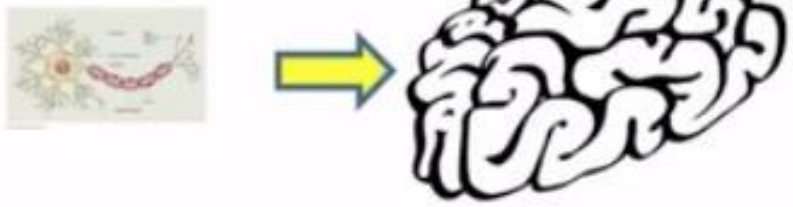


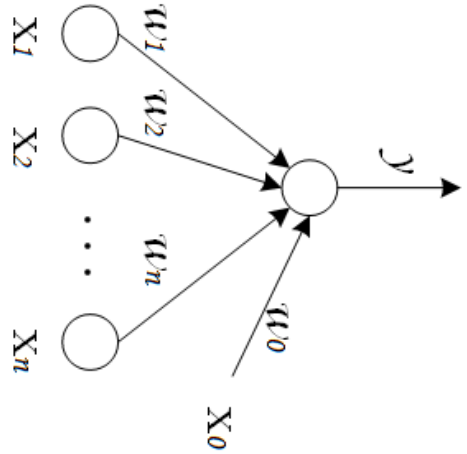


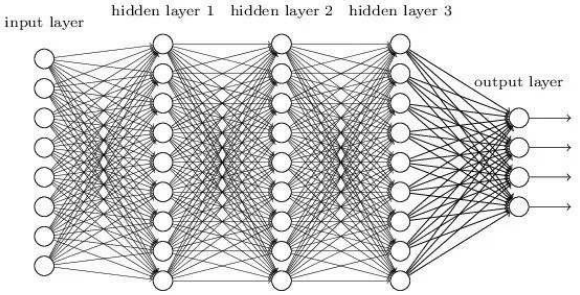


人工神经网络无法解决异或问题！！！

一个神经元不够



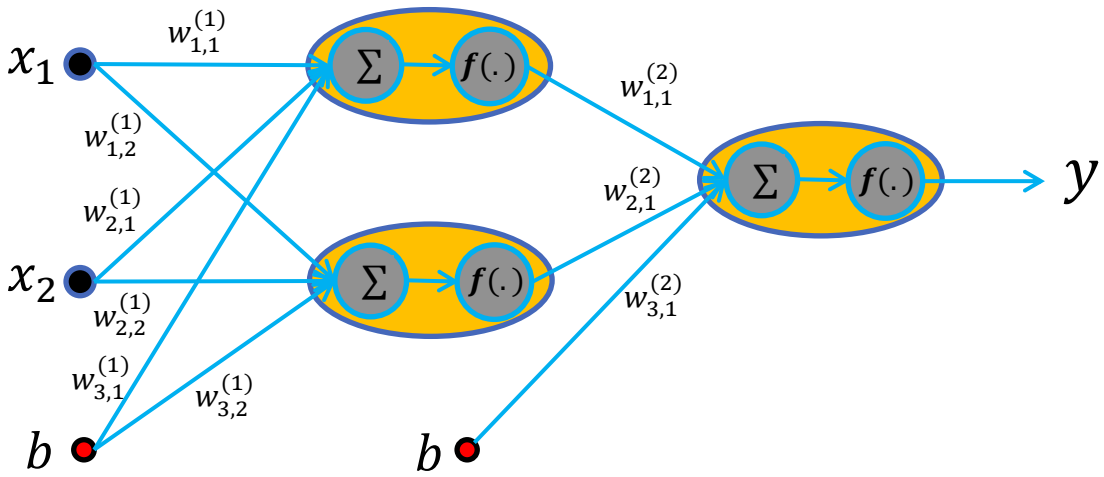


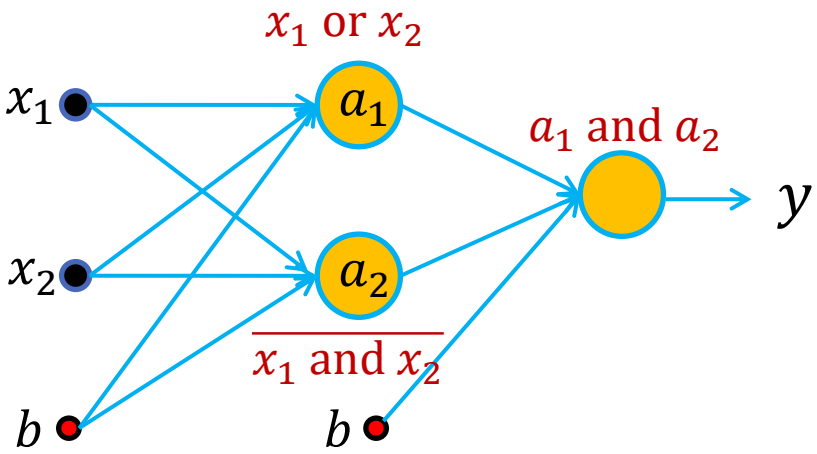


单一神经元的计算能力很弱

[1] Minsky M, Papert S A. Perceptrons: An introduction to computational geometry[M]. MIT press, 2017.

两层神经网络解决异或问题





隐藏层第一个神经元 $f(x_1 w_{1,1}^{(1)} + x_2 w_{2,1}^{(1)} + b w_{3,1}^{(1)})$

隐藏层第二个神经元 $f(x_1 w_{1,2}^{(1)} + x_2 w_{2,2}^{(1)} + b w_{3,2}^{(1)})$

真值表

x_1	x_2	a_1	a_2	y
0	0	**0**	**1**	0
0	1	**1**	**1**	1
1	0	**1**	**1**	1
1	1	**1**	**0**	0

异或运算真值表

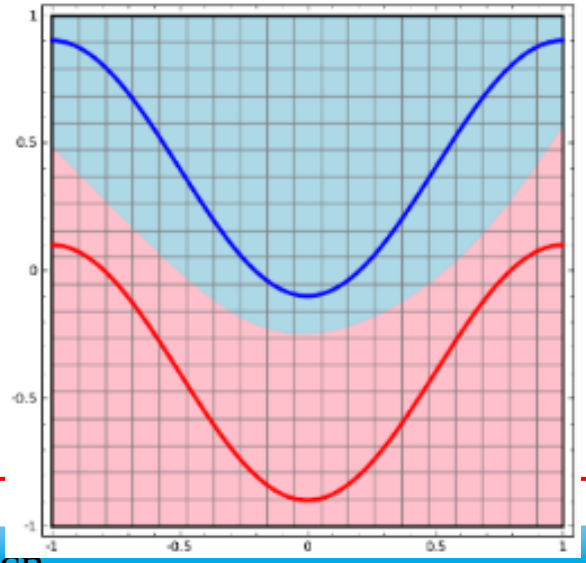
x_1	x_2	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

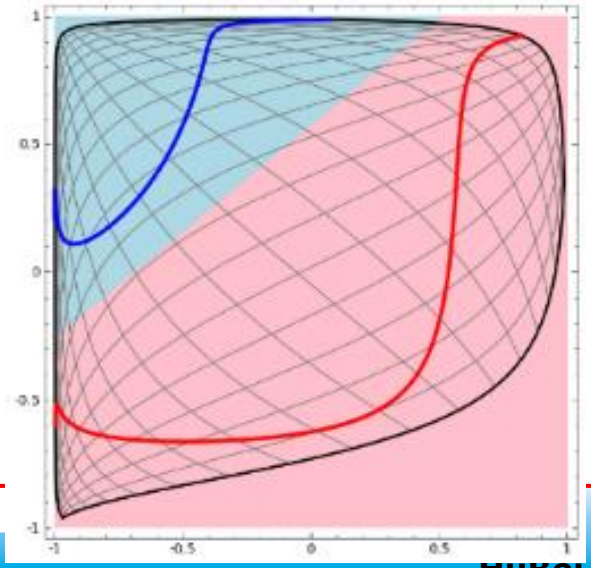
$$y = f(\ f(x_1 w_{1,1}^{(1)} + x_2\, w_{2,1}^{(1)} + \ b w_{3,1}^{(1)})\, w_{1,1}^{(2)} + f(x_1 w_{1,2}^{(1)} + x_2\, w_{2,2}^{(1)} + \ b w_{3,2}^{(1)})\, w_{2,1}^{(2)} + b w_{3,1}^{(2)})$$

1986年，反向传播算法（BP）被提出

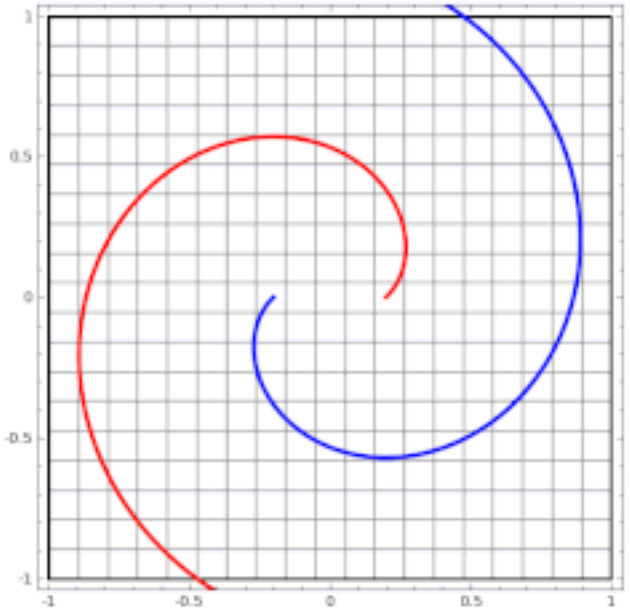
- Rumelhar和Hinton提出BP算法
  - 解决异或等分线性分类问题
  - 解决神经网络复杂计算的问题
  - 两层神经网络开始成为热点

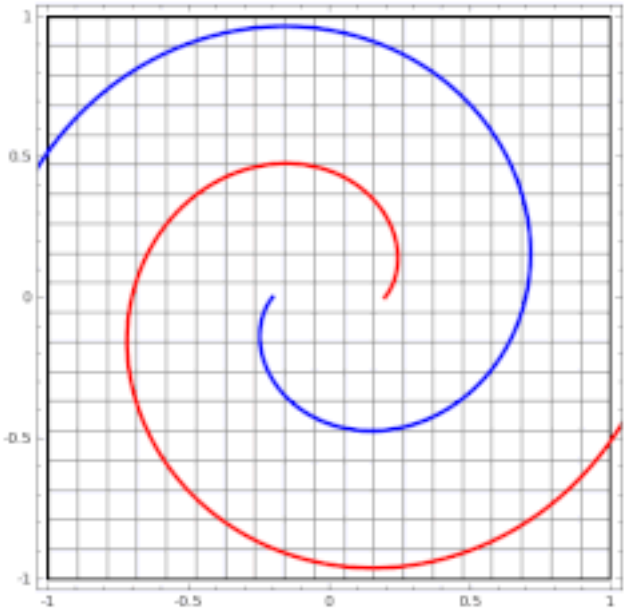
- 两层神经网络可以无限逼近任意连续函数





神经网络解决非线性二分类问题



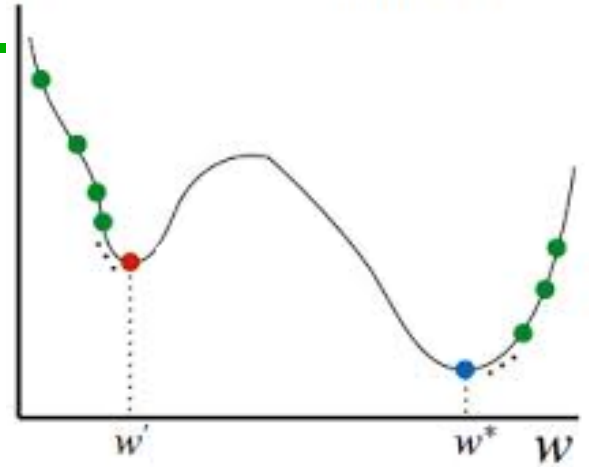


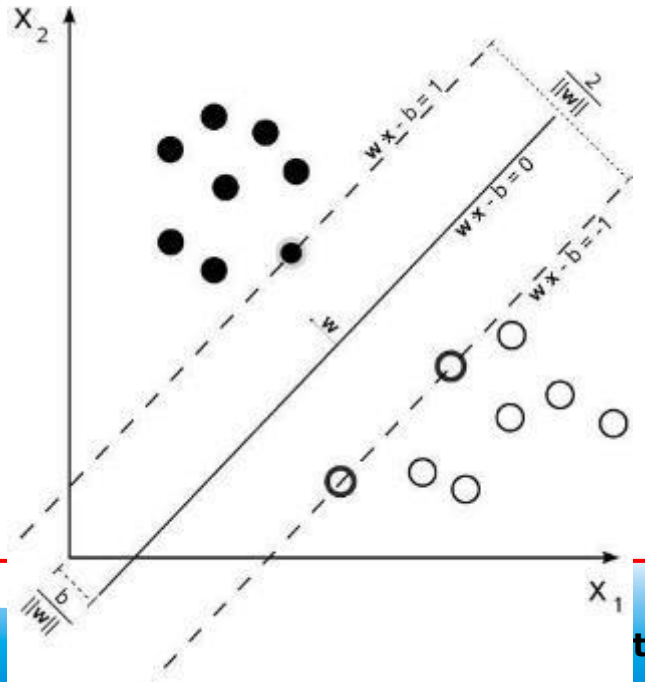
https://cs.stanford.edu/people/karpathy/convnetjs//demo/classify2d.html

20 世纪 90 年代中期神经网络再次没落

- 两层神经网络面临的问题
  - 局部最优
  - 调参繁琐

- 非神经网络算法逐步流行
  - Boosting
  - 支持向量机
  - 。。。



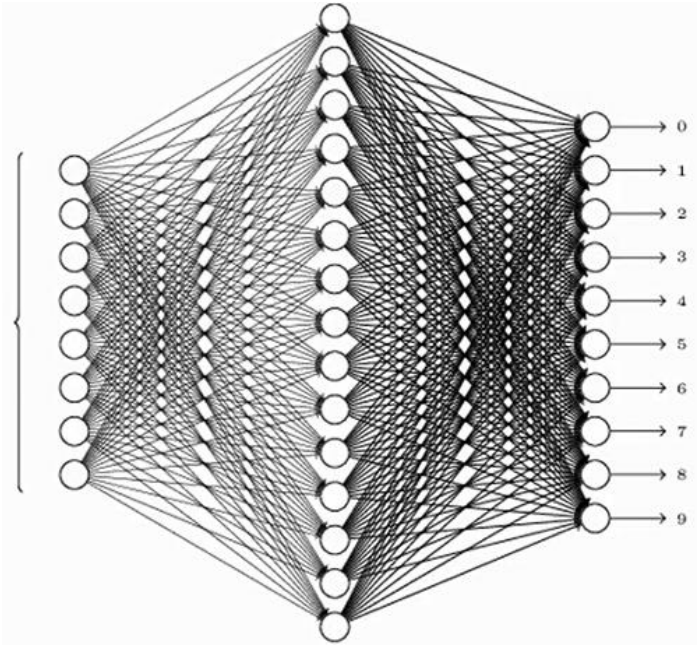


2006年开始，深度神经网络持续风靡

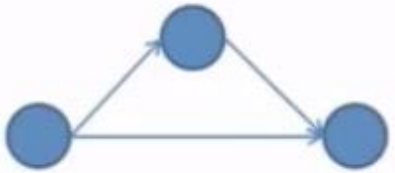
- Hinton在《Science》上发表文章提出深度学习
  - 提出“深度信念网络”
  - 提出“预训练”
  - 多层神经网络训练时间大幅减少

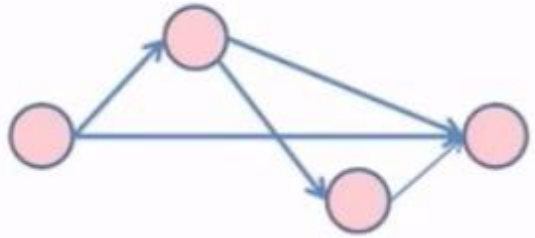
- 深度神经网络渗透并颠覆了诸多传统行业
  - 图像识别：CNN
  - 自然语言处理、语音识别：RNN
  - 。。。

深度神经网络



“深度”神经网络





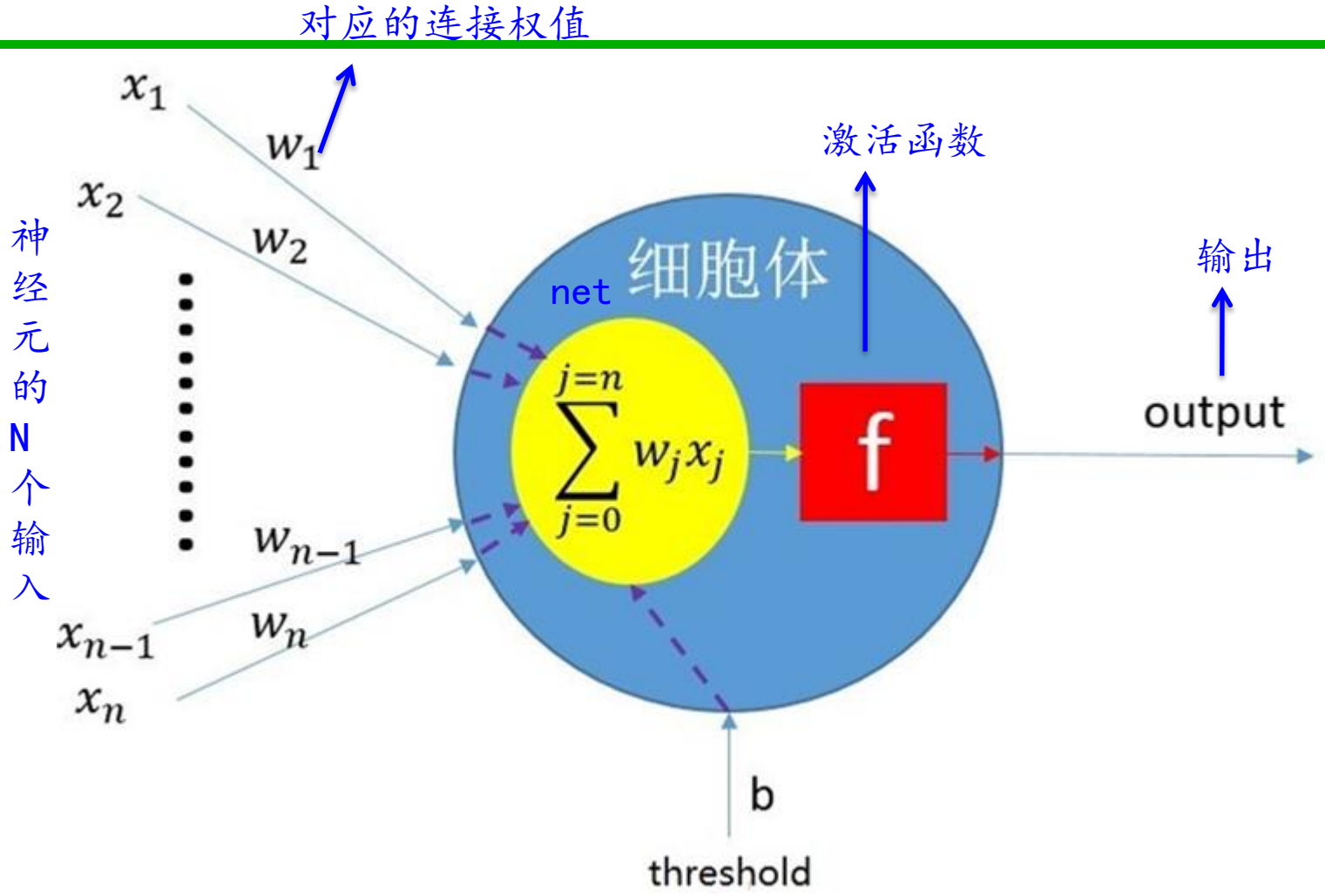
- Left: Depth = 2. Right: Depth = 3

要求

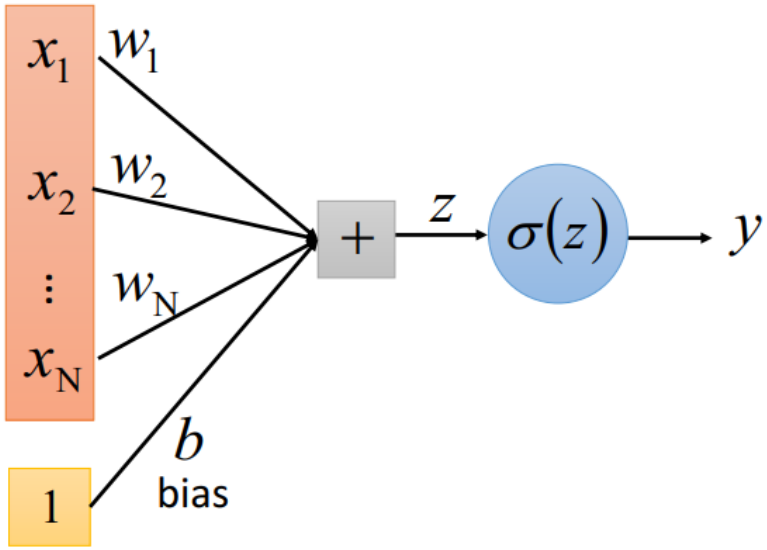
- **每一层捕获不同模型**
- **不丢失信息**

神经网络与MLP

基本人工神经元结构


对应的连接权值
x_1
w_1
x_2
w_2
神经元的N个输入
w_{n-1}
x_{n-1}
w_n
x_n
net
细胞体
$\sum_{j=0}^{j=n} w_j x_j$
激活函数
f
输出
output
b
threshold

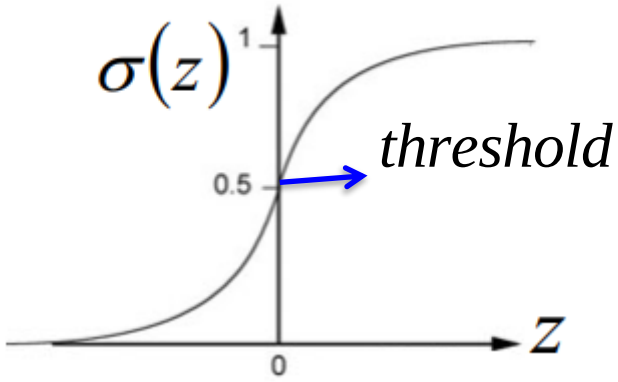
基本人工神经元结构



$$z = w_1 x_1 + w_2 x_2 + \ldots + w_N x_N + b$$

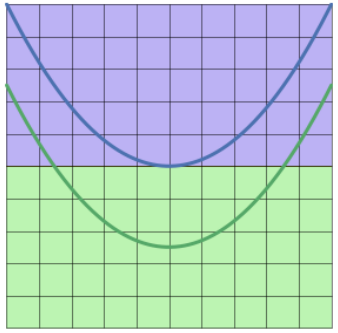
$$y = \begin{cases} 1 & a \geq threshold \\ 0 & a < threshold \end{cases}$$

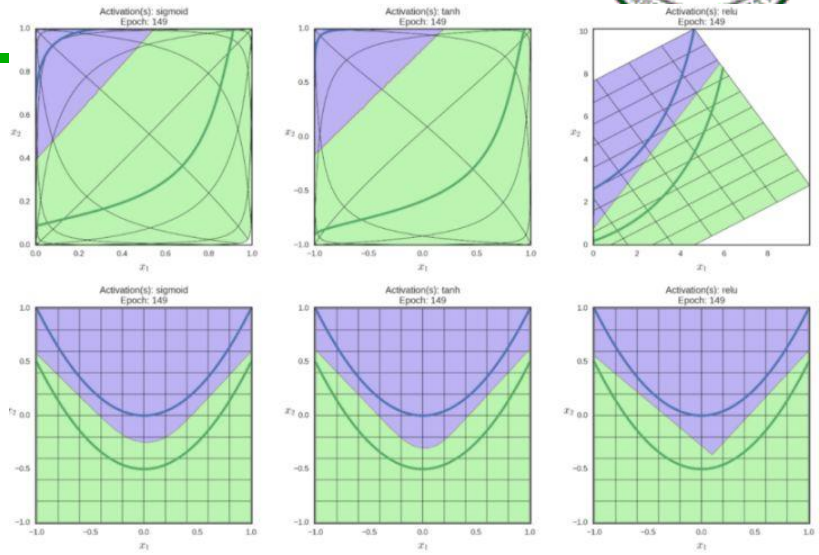
a



激活函数 — 引入非线性因素

- 激活函数可能具有的特性
  - h: R->R 且 几乎处处可导
    - 反向传播时需要计算激活函数的偏导数
  - 非线性函数
    - 在神经网络中引入非线性因素
  - 单调性
    - 当激活函数是单调的时候，单层网络能够保证是凸函数
  - 输出值范围
    - 当激活函数输出值是 有限 的时候，基于梯度的优化方法会更加稳定，因为特征的表示受有限权值的影响更显著；
    - 当激活函数的输出是 无限 的时候，模型的训练会更加高效，不过在这种情况小，一般需要更小的learning rate



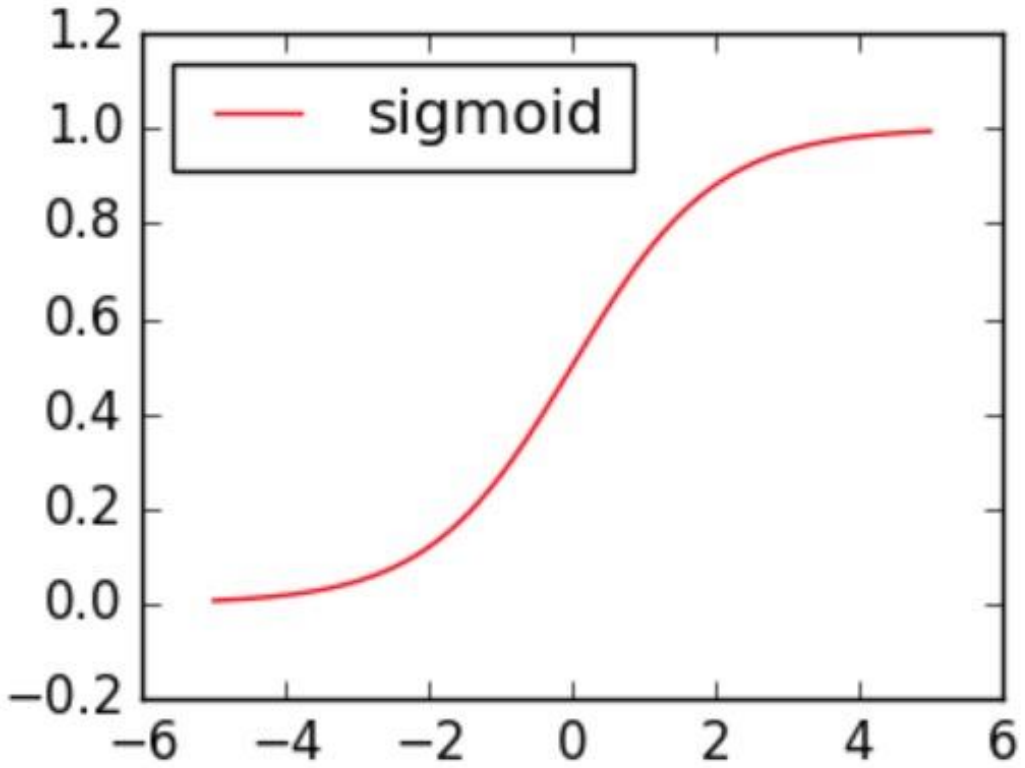


Hyper-parameters in Action!
— Daniel Godoy

常见的激活函数

- Sigmoid函数

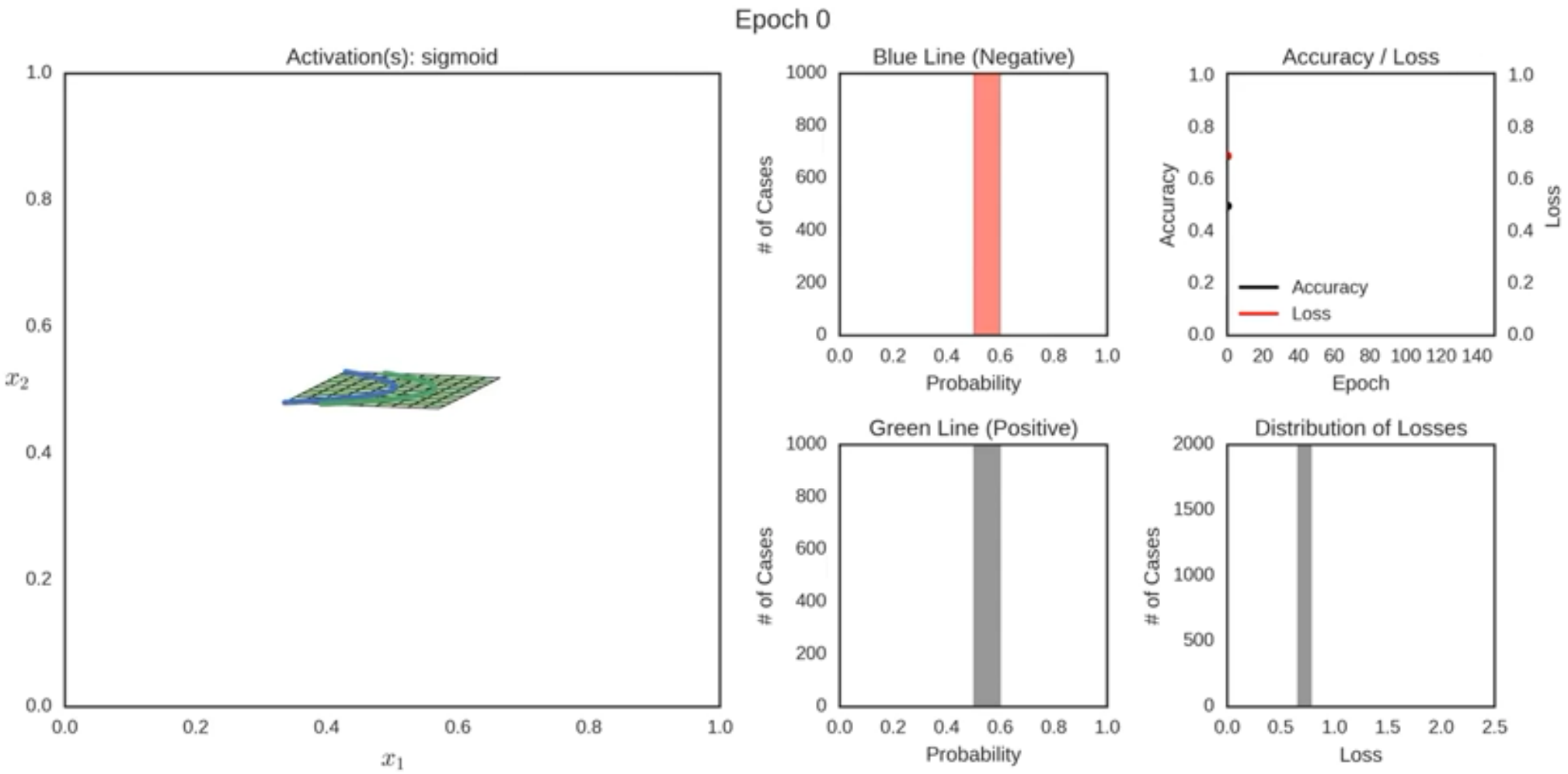
$$f(x) = \frac{1}{1 + e^{-x}}$$



- 优点
  - 具有较好的可解释性
  - 输输出范围有限
  - 单调连续

- 缺点
  - 函数饱和，梯度容易消失
  - 输出不以0为中心

Sigmoid

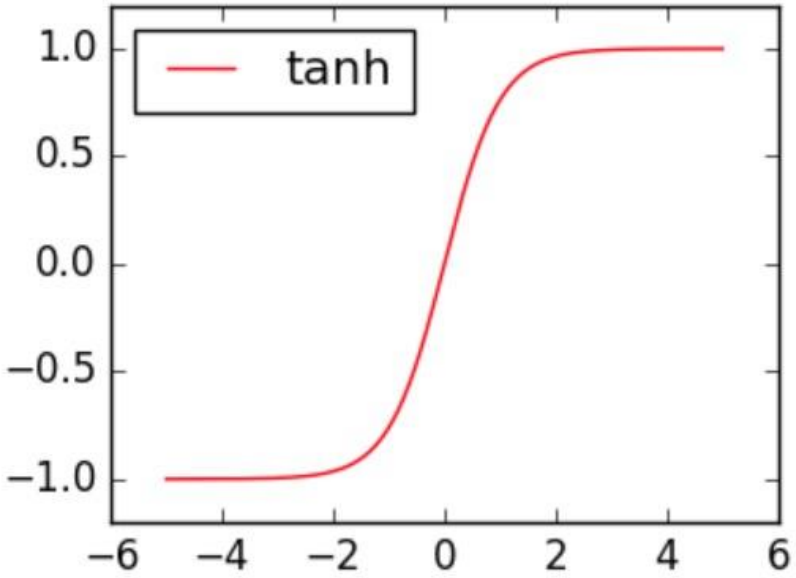


Hyper-parameters in Action!
— Daniel Godoy

常见的激活函数

- Tanh函数

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

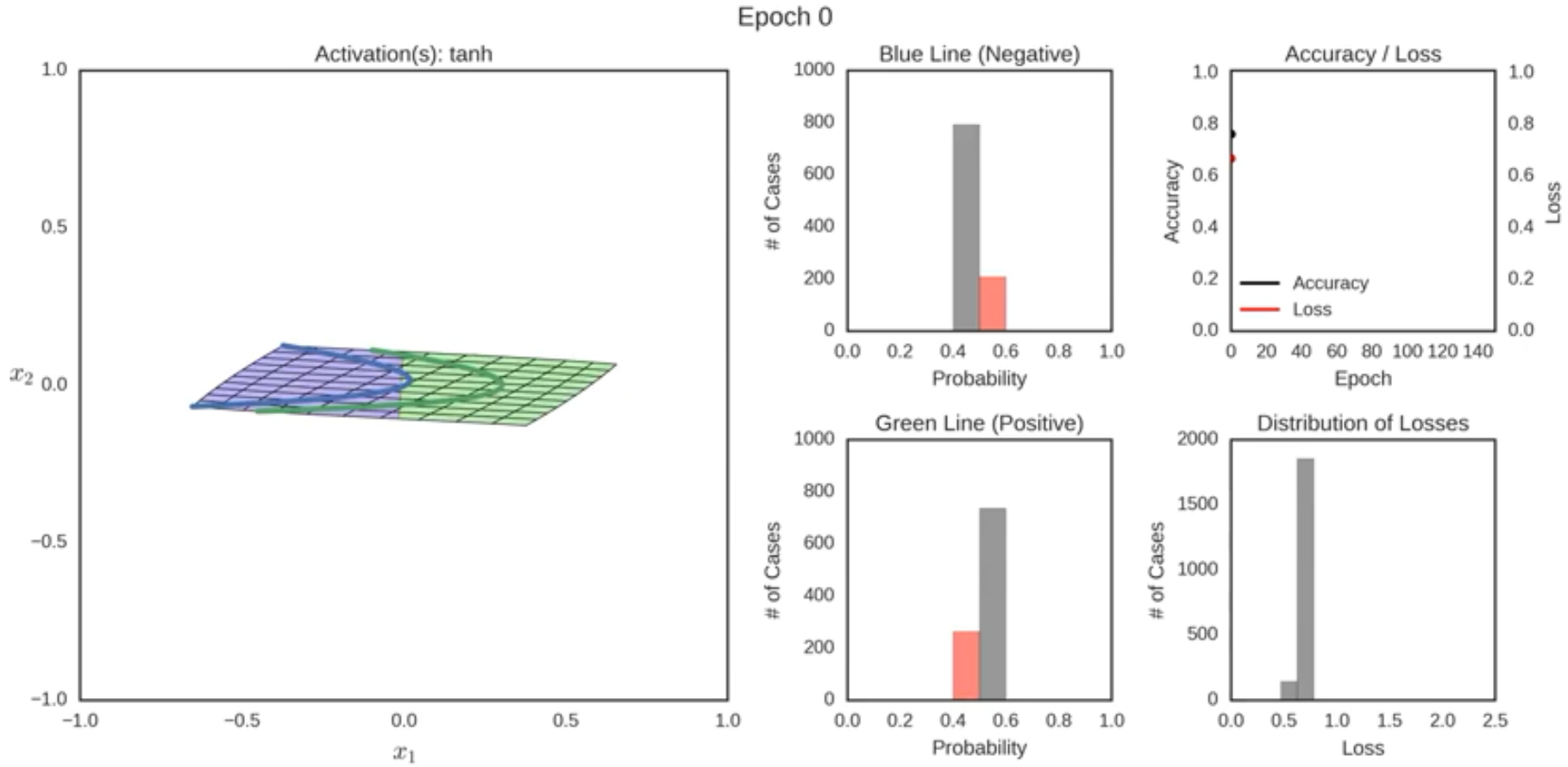


- 优点
  - 比Sigmoid收敛速度快
  - 输出以0为中心

- 缺点
  - 函数饱和，梯度容易消失

$$tanh(x) = 2\,sigmoid(2x) - 1$$

Tanh

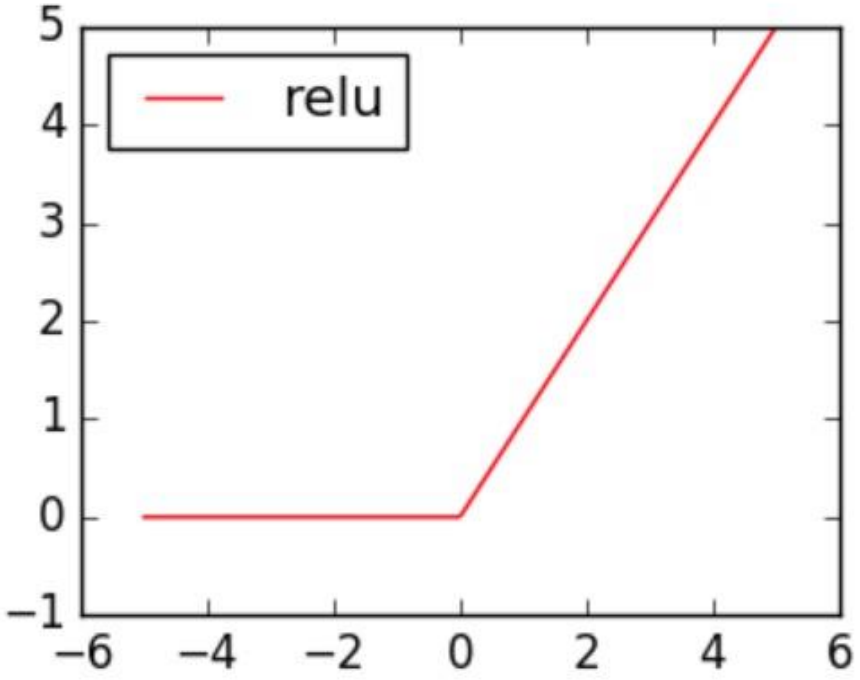


Hyper-parameters in Action!
— Daniel Godoy

常见的激活函数

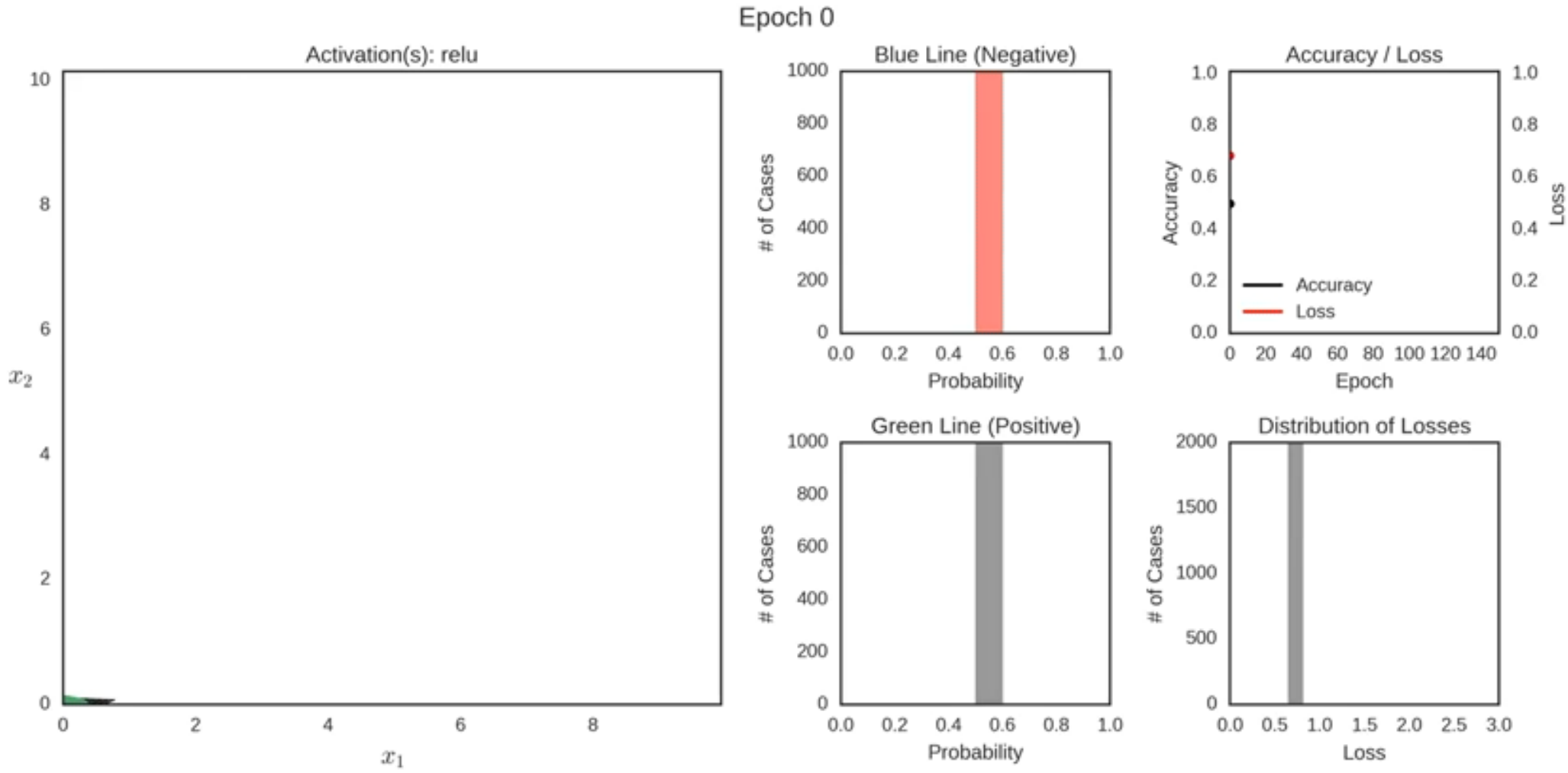
- ReLU（Rectified Linear Units）函数

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



- 优点
  - 收敛速度比上述激活函数更快
  - 计算简单
- 缺点
  - 当输入小于0时，训练参数将无法更新

ReLU

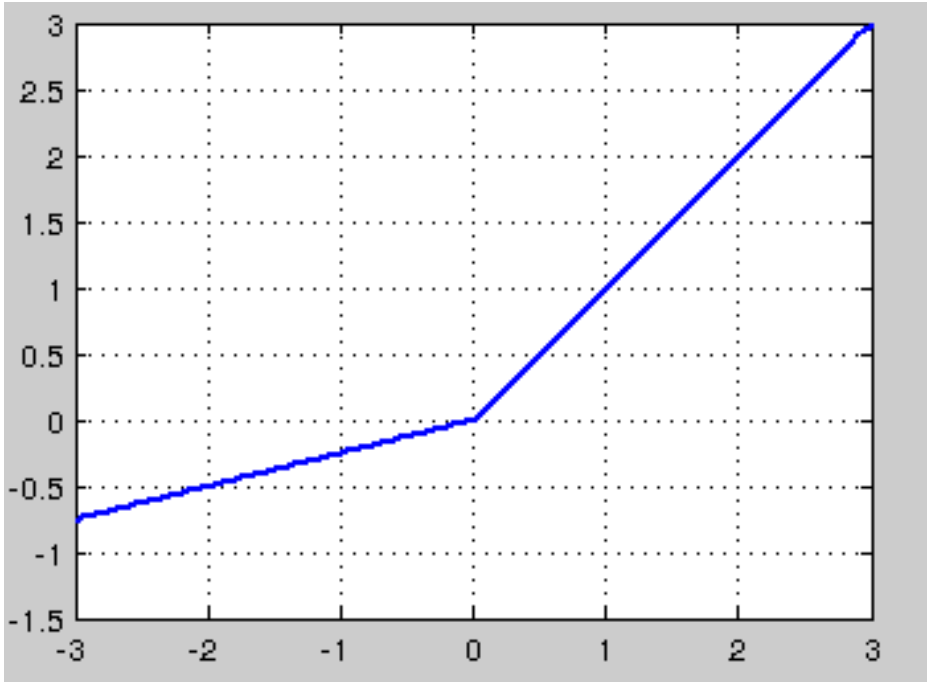


Hyper-parameters in Action!
— Daniel Godoy

常见的激活函数

- Leaky-ReLU函数

$$f(x)=\begin{cases} x, & if\ x \geq 0 \\ \alpha x, & if\ x < 0 \end{cases}$$



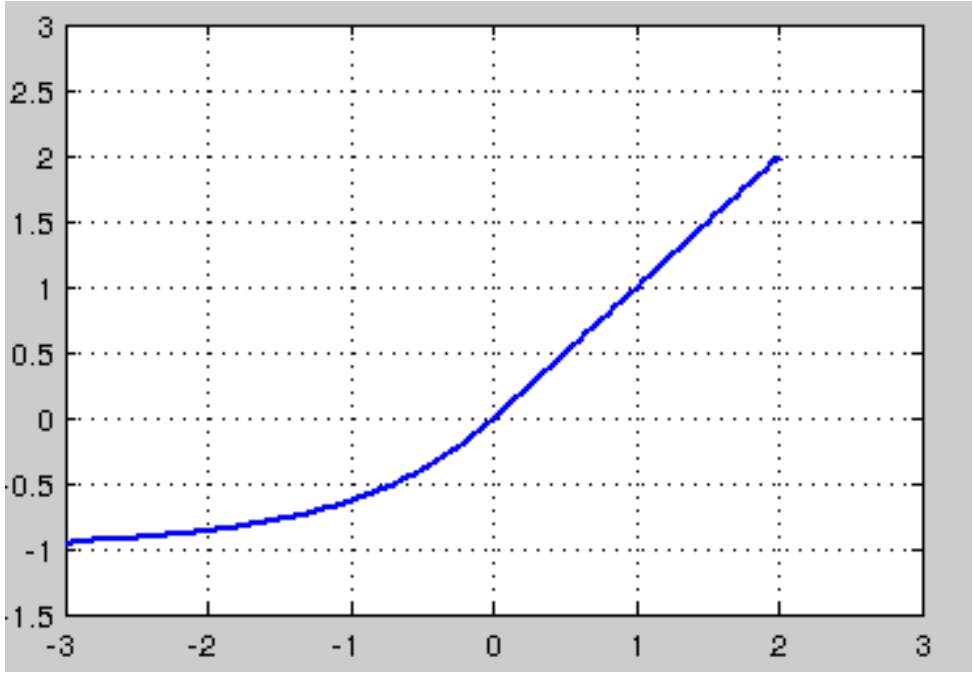
- 优点
  - 收敛速度快
  - 解决神经元死亡现象
- 缺点
  - 实际应用中发现效果不稳定

常见的激活函数

- ELU函数

$$f(x)=\begin{cases} x, & if\ x\geq 0 \\ \alpha\left(e^{x}-1\right), & if\ x<0 \end{cases}$$

- 优点
  - 收敛速度快
  - 缓解梯度消失
  - 对输入变化更鲁棒

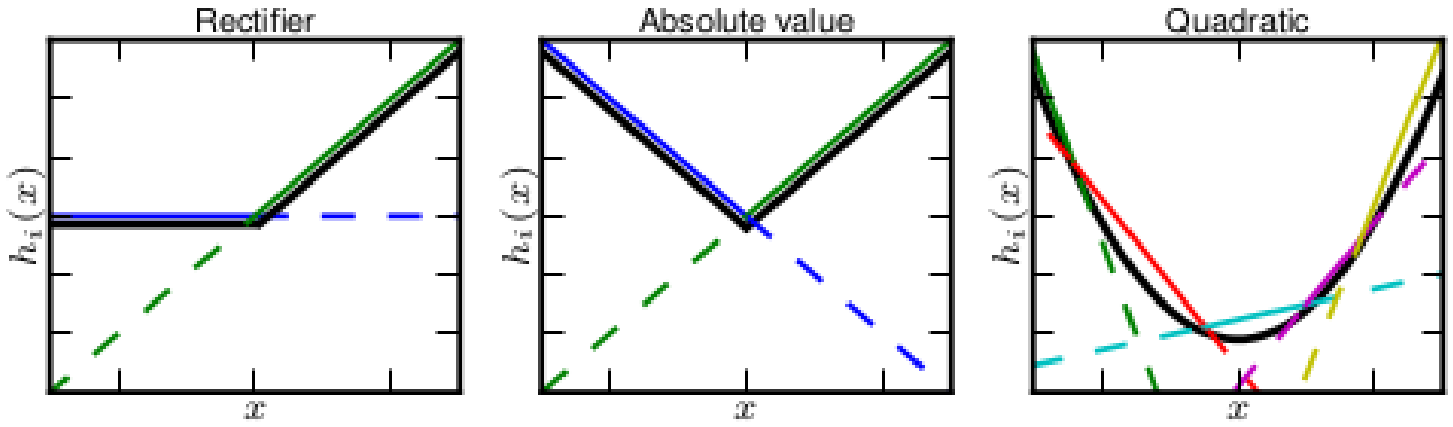


常见的激活函数

- Maxout函数

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2, \cdots, w_n^T + b_n)$$

- 优点
  - 收敛速度快
  - 不饱和
  - 可以拟合任意凸函数

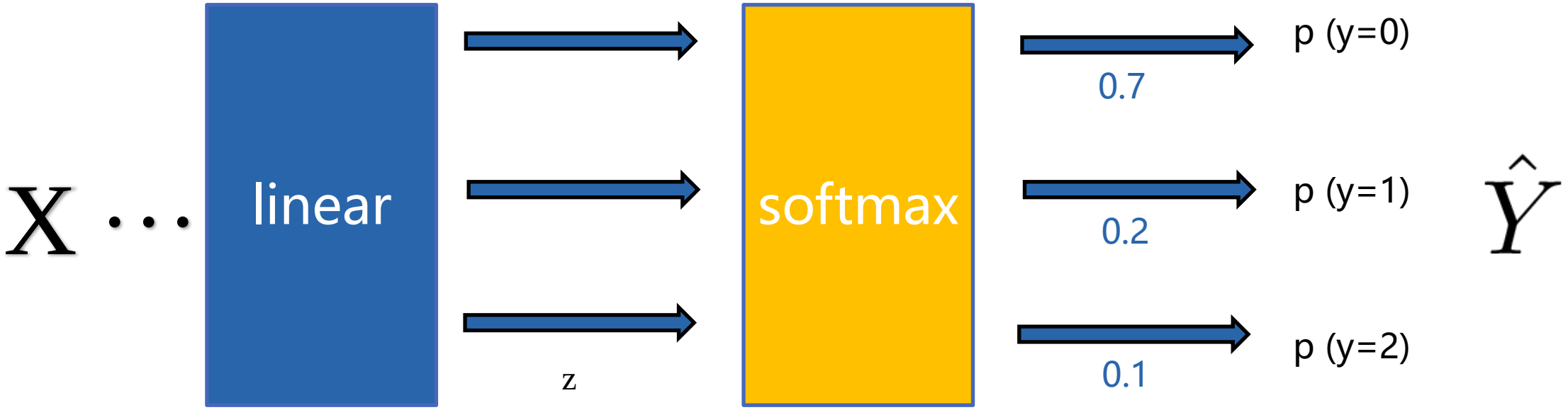


- 缺点
  - 参数量增加，计算复杂度增大

常见的激活函数

- Softmax 函数将多个神经元输出映射到区间(0, 1)， 常用于多分类问题中

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_i e^{z_i}}$$



多层感知机

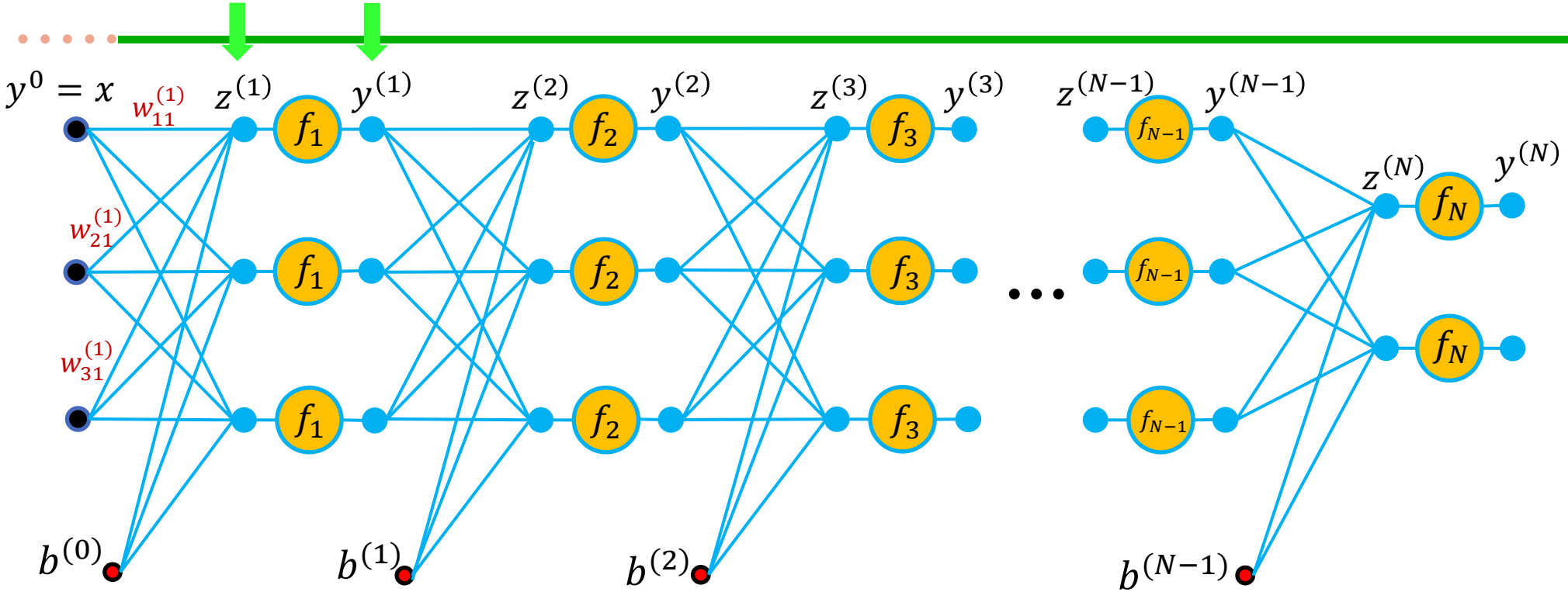
- 多层感知机模型是一种前馈神经网络
  - 前馈 ⟶ 信息从输入层到输出层单向流动

- 多层感知机模型是一个监督学习模型
  - 通过在训练集中寻找最优参数$\hat{\theta}$，从而得到函数$f(x,\theta)$ 的最优近似$f(x,\hat{\theta})$

- 将含有一层隐含层的感知机网络推广到多层网络结构，数学表达式为

$$f(x) = f^{(n)}(f^{(n-1)}(\dots f^{1}(x)))$$

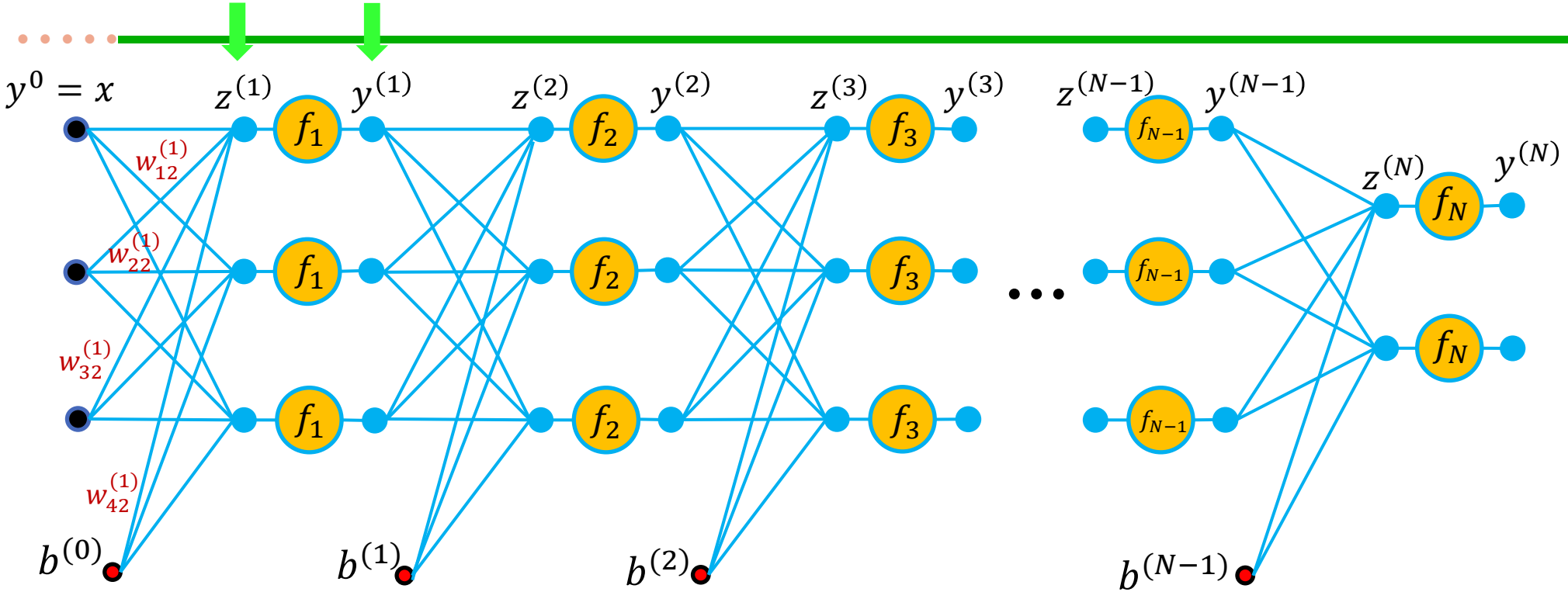
多层神经网络模型



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b^{(0)}$$

$$y_j^{(1)} = f_1(z_j^{(1)})$$

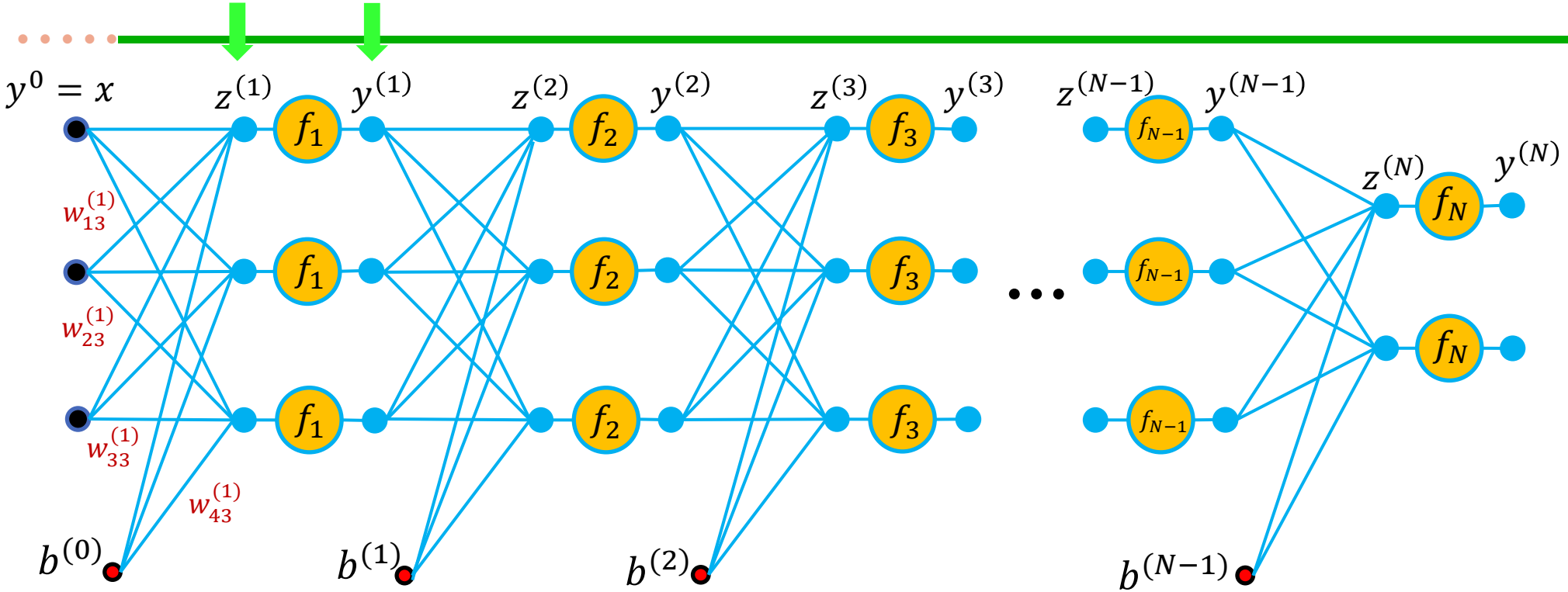
多层神经网络模型



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b^{(0)}$$

$$y_j^{(1)} = f_1(z_j^{(1)})$$

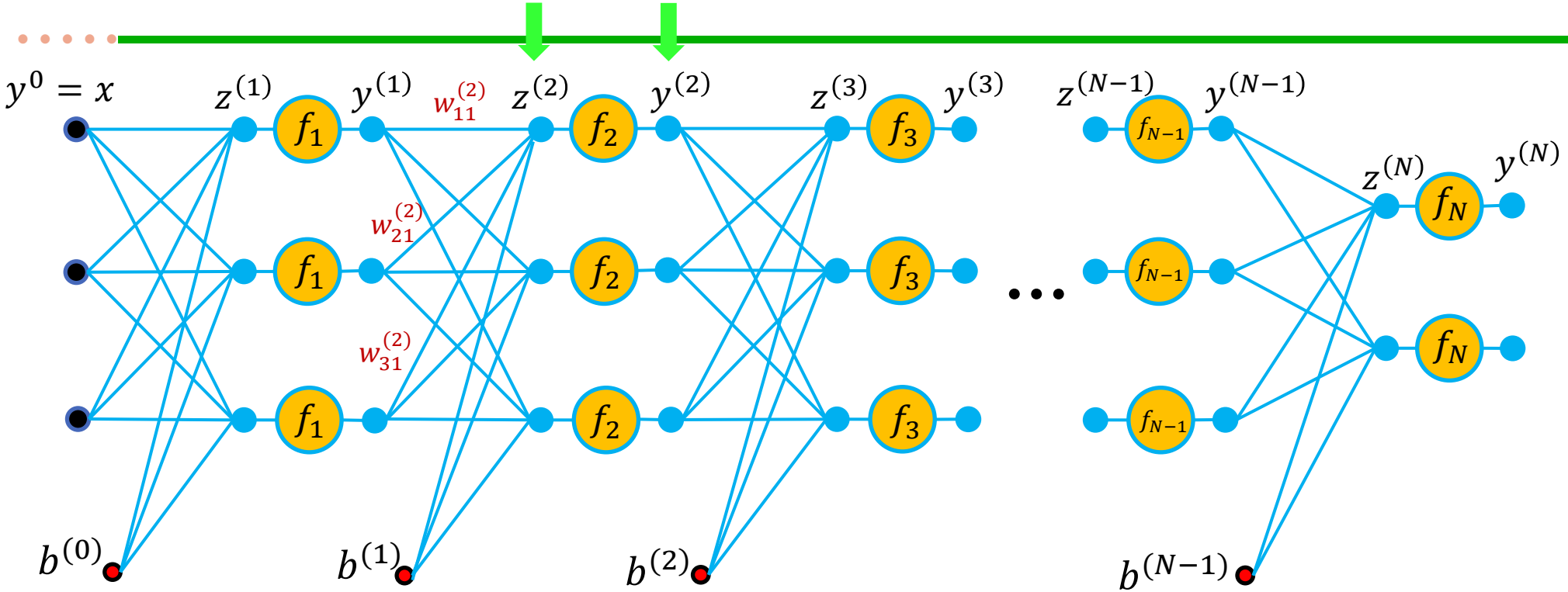
多层神经网络模型



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b^{(0)}$$

$$y_j^{(1)} = f_1(z_j^{(1)})$$

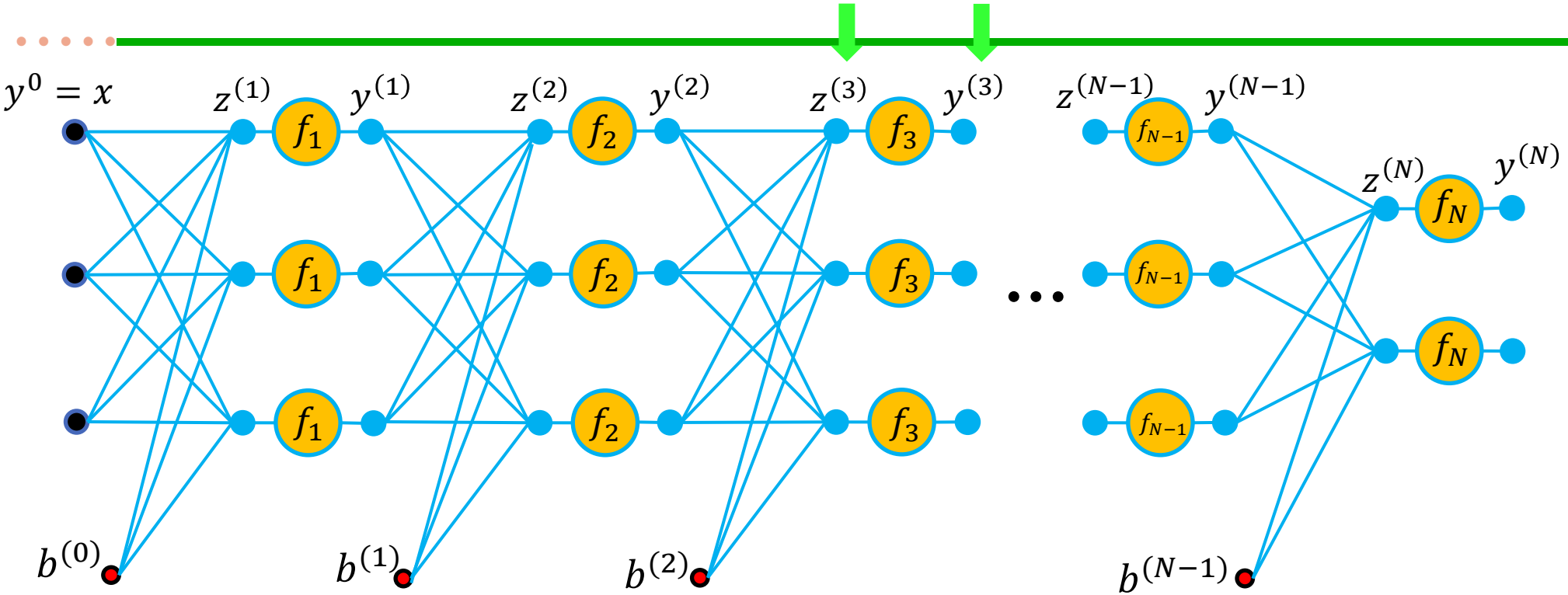
多层神经网络模型



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b \qquad z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^1 + b^{(1)}$$

$$y_j^{(1)} = f_1(z_j^{(1)}) \qquad y_j^{(2)} = f_2(z_j^{(2)})$$

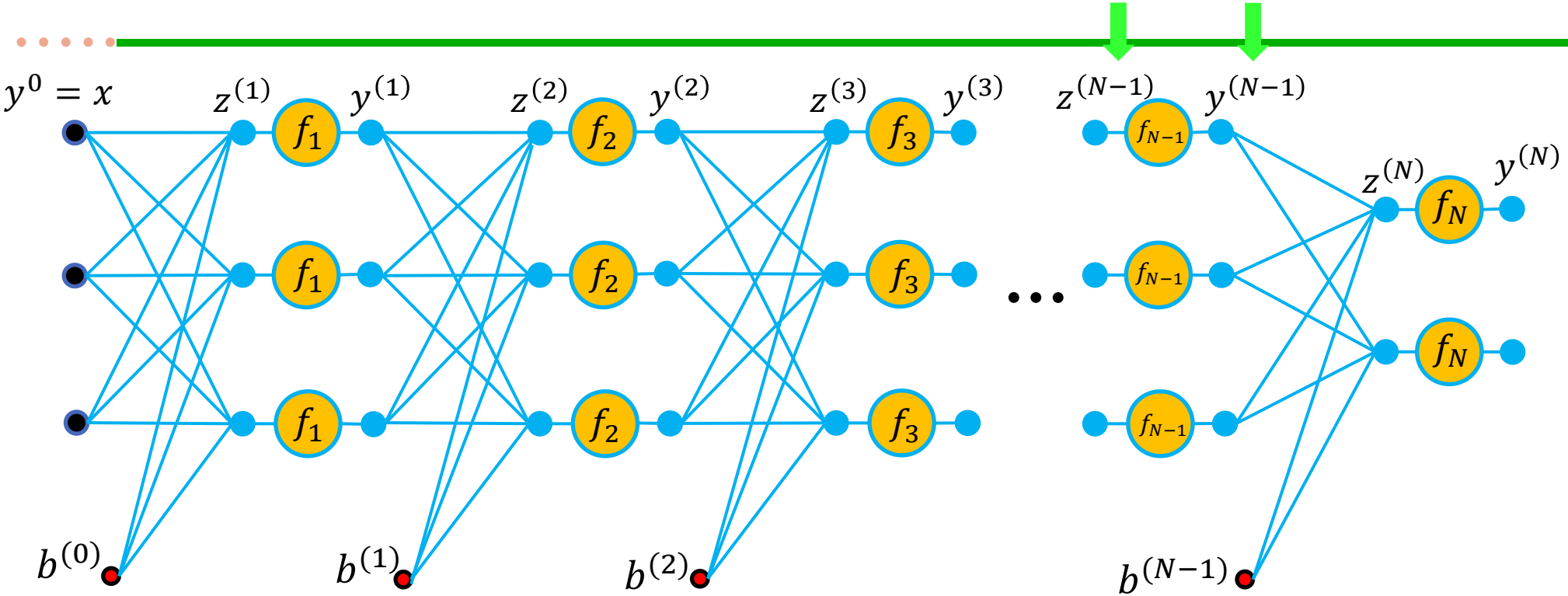
多层神经网络模型



$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b^{(0)} \quad z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^1 + b^{(1)} \quad z_j^{(3)} = \sum_i w_{ij}^{(3)} y_i^2 + b^{(2)} \quad \cdots$$

$$y_j^{(1)} = f_1(z_j^{(1)}) \qquad y_j^{(2)} = f_2(z_j^{(2)}) \qquad y_j^{(3)} = f_2(z_j^{(3)})$$

多层神经网络模型

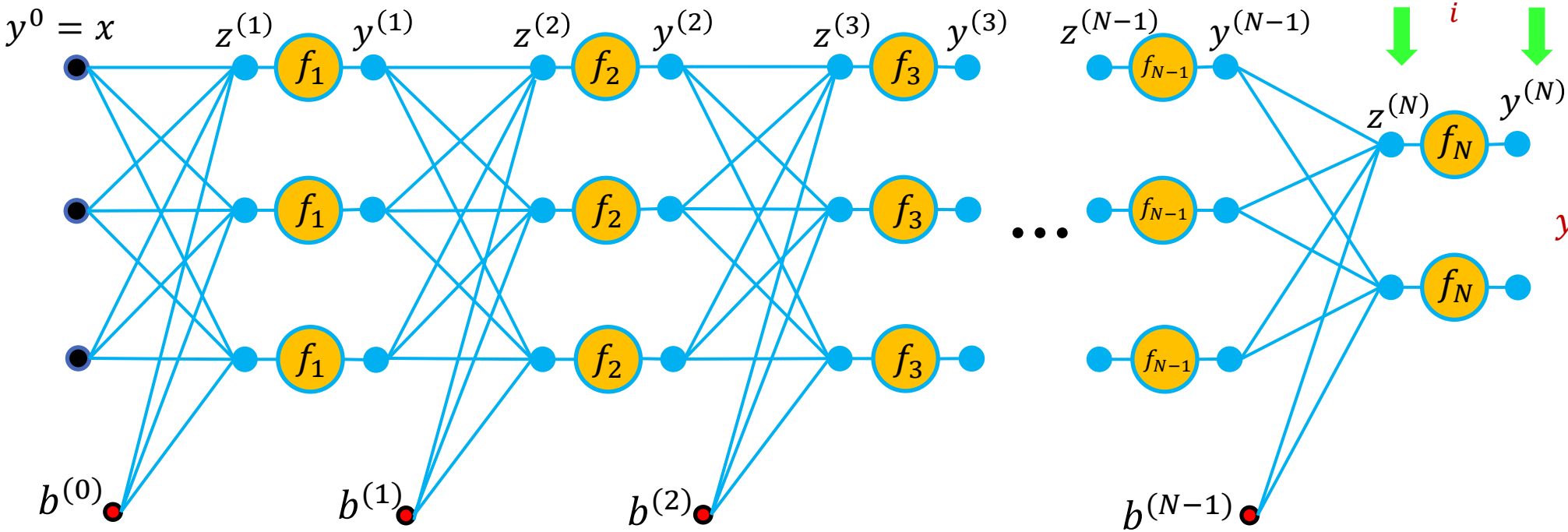


$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b^{(0)} \quad z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^1 + b^{(1)} \quad z_j^{(3)} = \sum_i w_{ij}^{(3)} y_i^2 + b^{(2)} \quad \cdots \quad z_j^{(N-1)} = \sum_i w_{ij}^{(N-1)} y_i^{(N-2)} + b^{(N-2)}$$

$$y_j^{(1)} = f_1(z_j^{(1)}) \quad y_j^{(2)} = f_2(z_j^{(2)}) \quad y_j^{(3)} = f_2(z_j^{(3)}) \quad \cdots \quad y_j^{(N-1)} = f_{N-1}(z_j^{(N-1)})$$

多层神经网络模型

$$z_j^{(N)} = \sum_i w_{ij}^{(N)} y_i^{(N-1)} + b^{(N-1)}$$

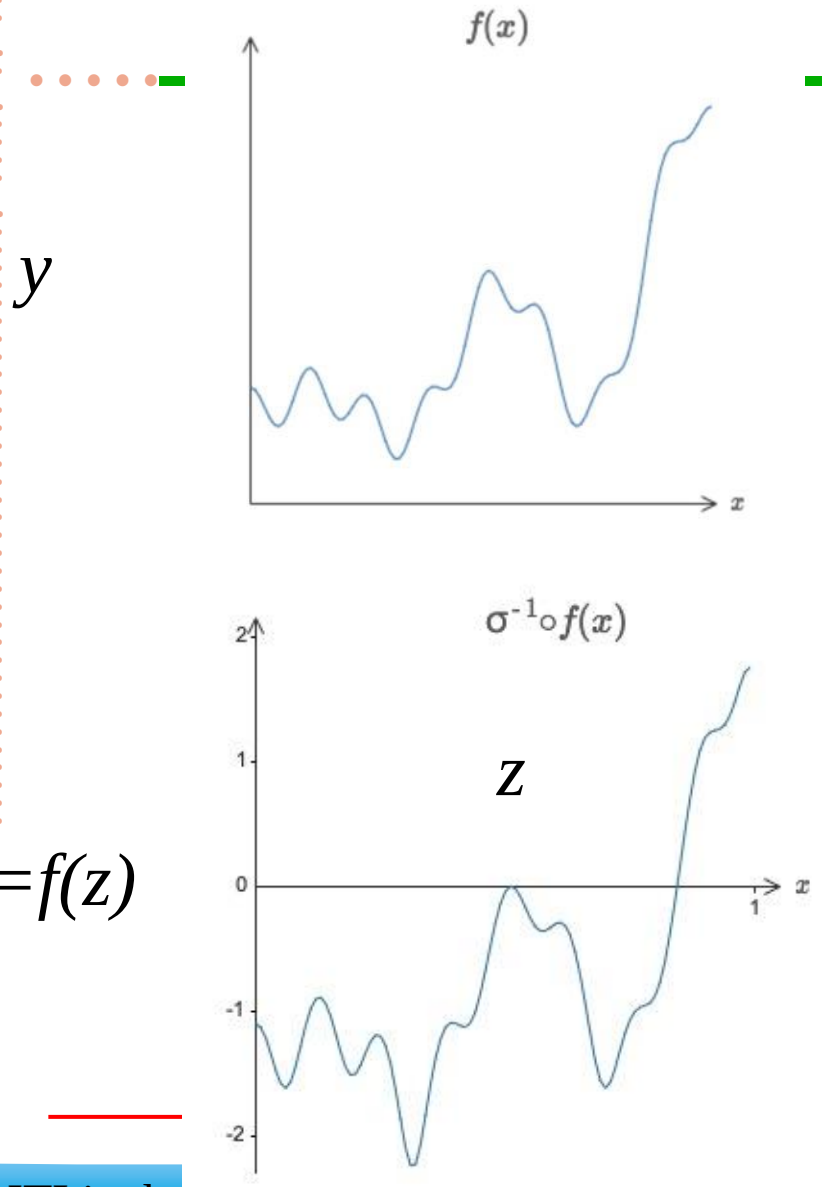


$$y_j^{(N)} = f_N(z_j^{(N)})$$

$$z_j^{(1)} = \sum_i w_{ij}^{(1)} y_j^0 + b^{(0)} \quad z_j^{(2)} = \sum_i w_{ij}^{(2)} y_i^1 + b^{(1)} \quad z_j^{(3)} = \sum_i w_{ij}^{(3)} y_i^2 + b^{(2)} \quad \cdots \quad z_j^{(N-1)} = \sum_i w_{ij}^{(N-1)} y_i^{(N-2)} + b^{(N-2)}$$

$$y_j^{(1)} = f_1(z_j^{(1)}) \quad y_j^{(2)} = f_2(z_j^{(2)}) \quad y_j^{(3)} = f_2(z_j^{(3)}) \quad \cdots \quad y_j^{(N-1)} = f_{N-1}(z_j^{(N-1)})$$

神经网络如何拟合一个一元函数



前馈神经网络

只需具备单层隐含层和有限个神经单元

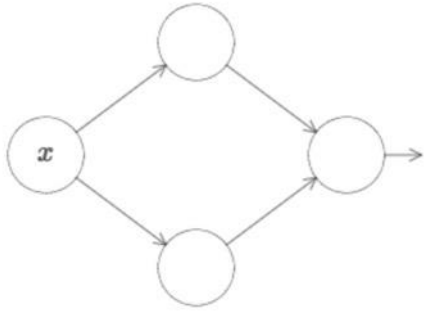
就能以任意精度拟合任意复杂度的函数

—— 万能近似定理 Universal approximation theorem

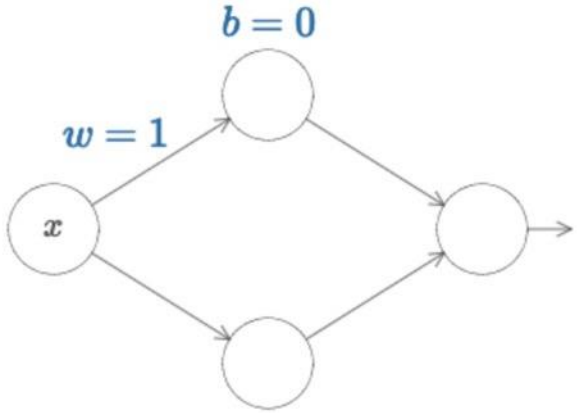
Hornik et al., 1989; Cybenko, 1989

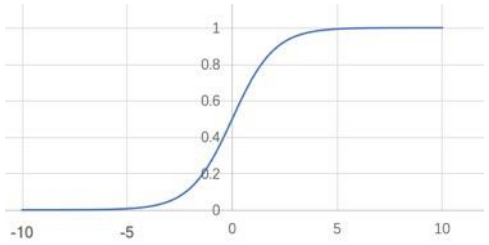
A visual proof that neural nets can compute any function
http://neuralnetworksanddeeplearning.com/chap4.html

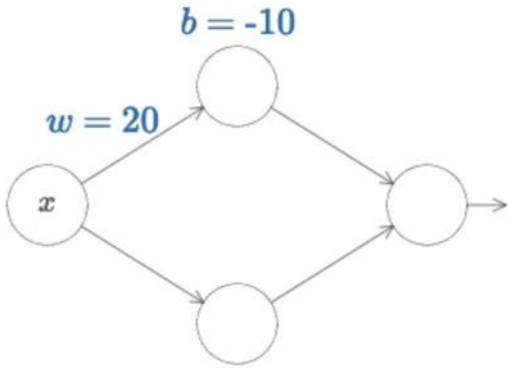
一个简单的双层神经网络

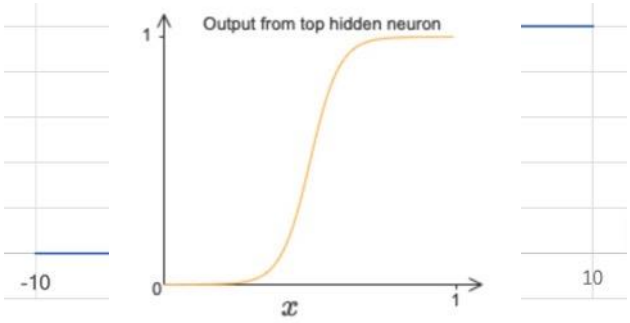


$$\sigma(z) = 1/(1+e^{-z}) \quad z = wx + b$$

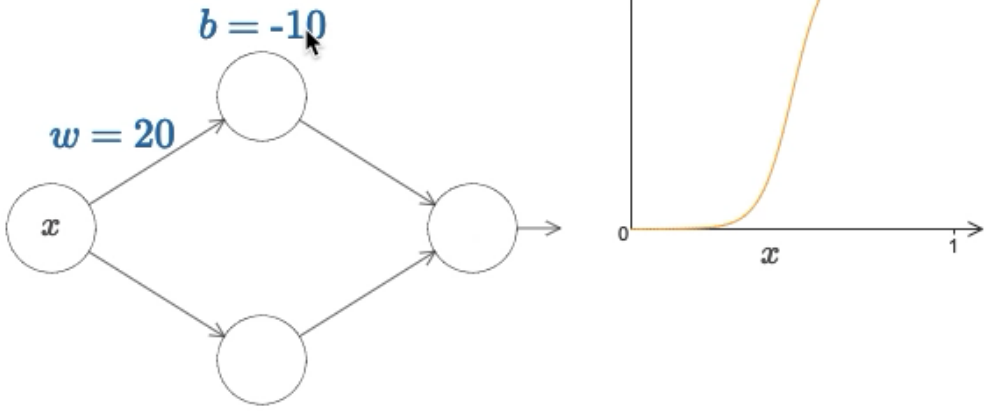




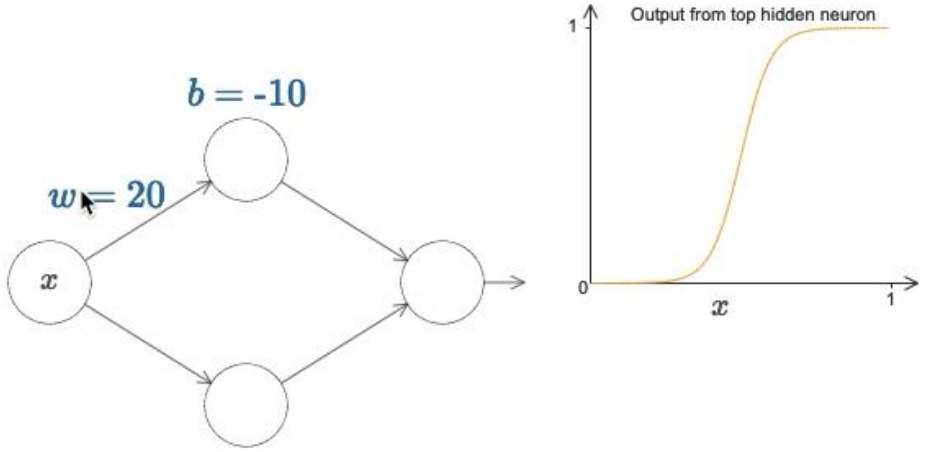




输入层到隐藏层的参数如何影响输出？—改变偏置/权重

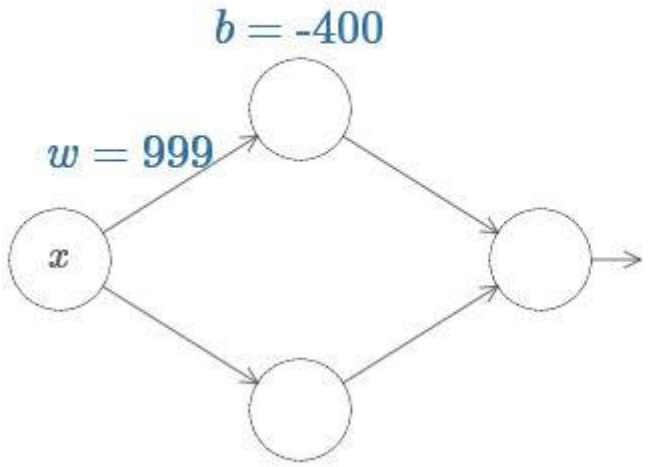


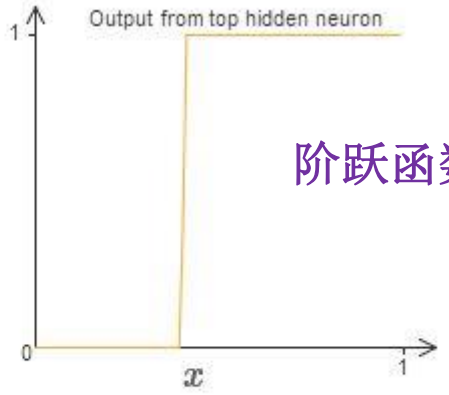
改变偏重



改变权值

输入层到隐藏层的参数如何影响输出？—改变偏置和权重

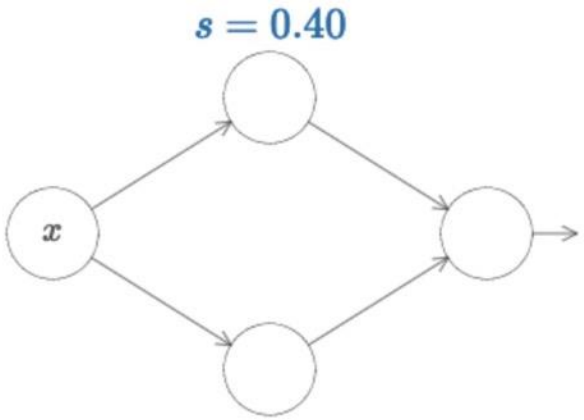


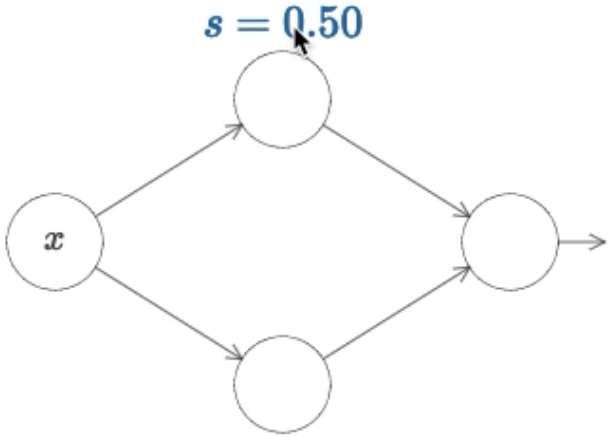


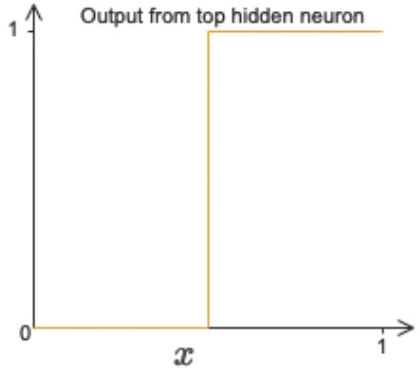
阶跃函数

s = -b/w

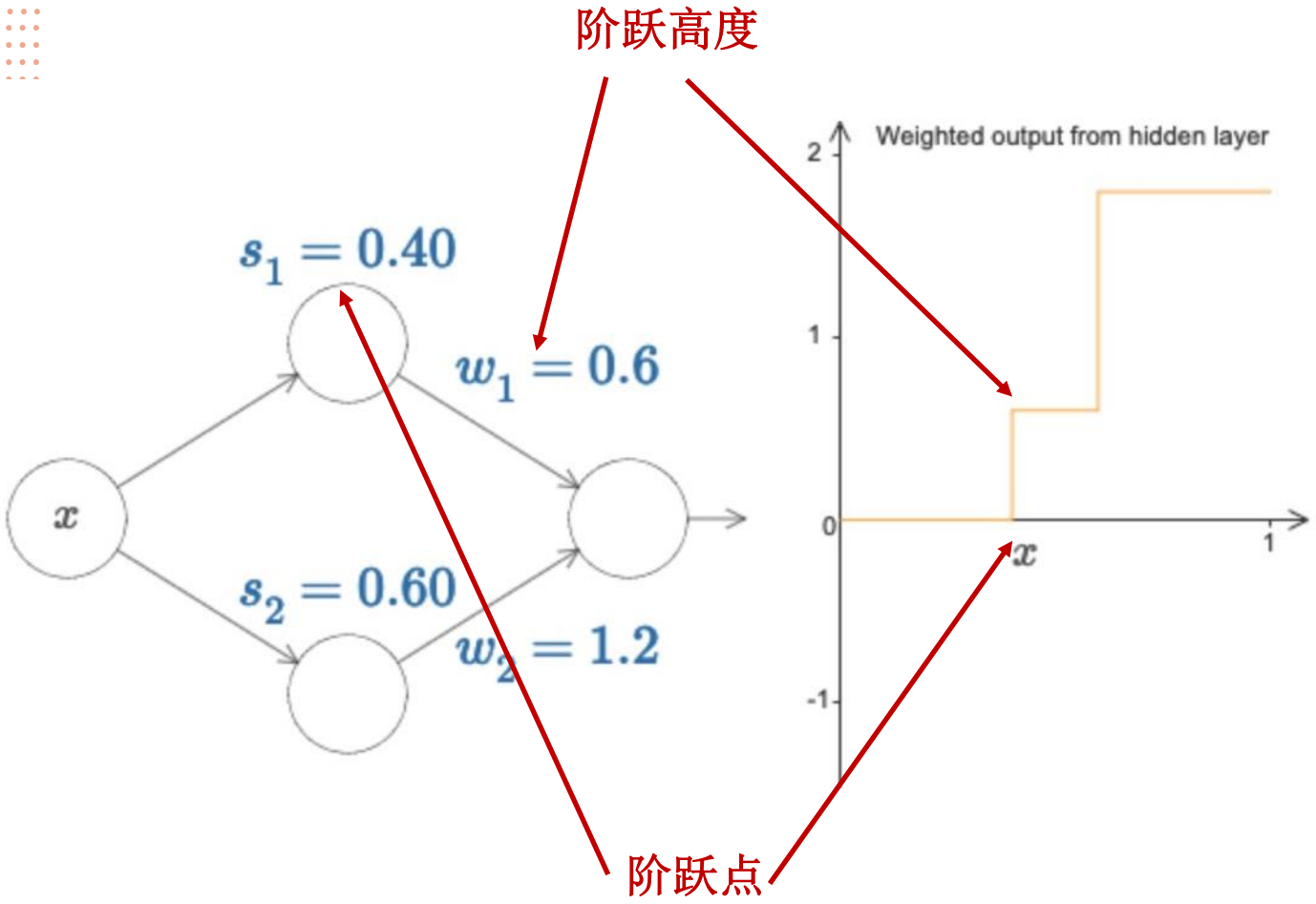
s 是阶跃函数的阈值







隐藏层到输出层的参数如何影响输出?



问题1

- s1 是负数会如何?

问题2

- s1 如果大于 s2，那么图形会怎样变化?

问题3

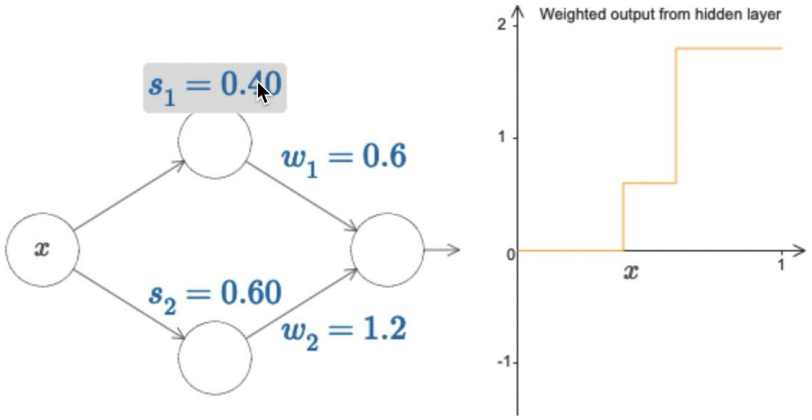
- w1 如果大于w2会如何?

问题4

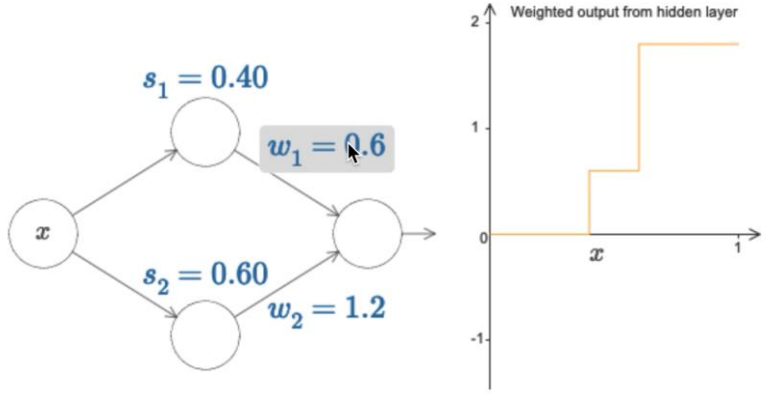
- w1 是负数会如何?

隐藏层神经元的相互作用

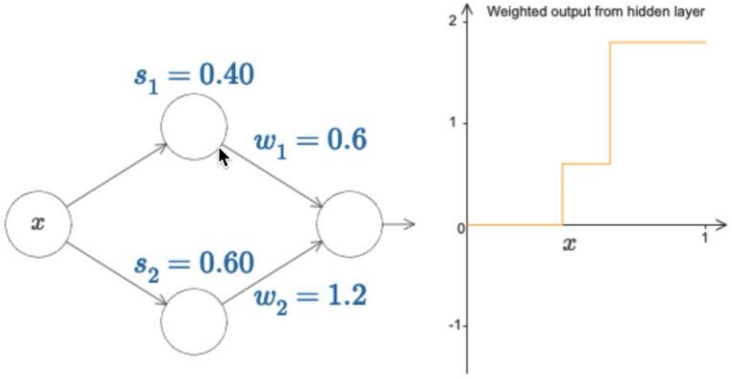
S1 变为负数



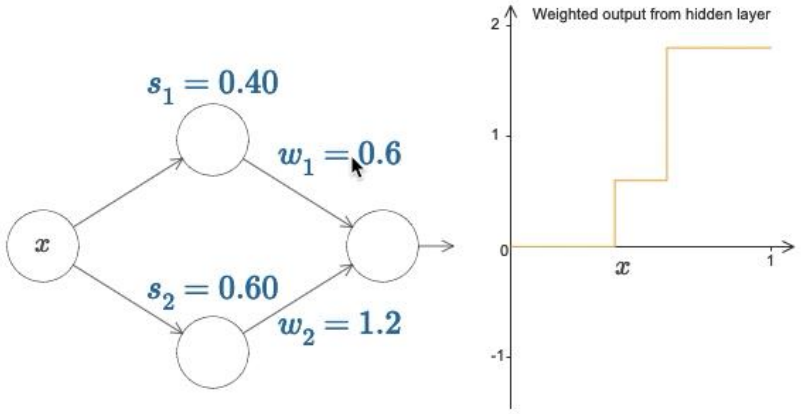
w1 大于 w2



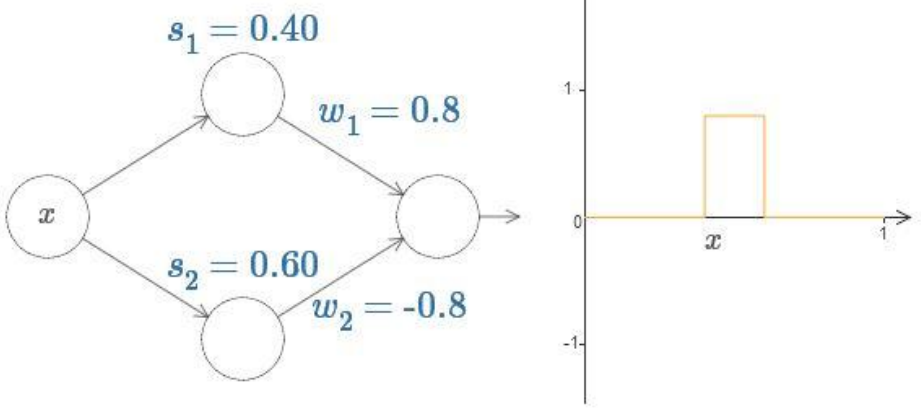
S1 大于 s2

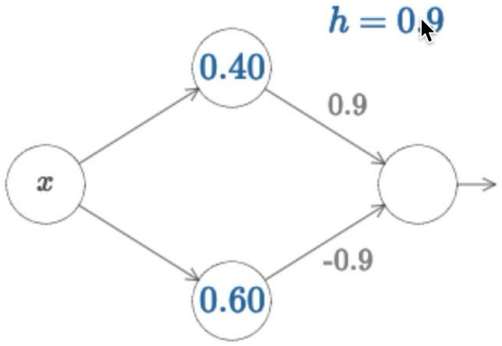


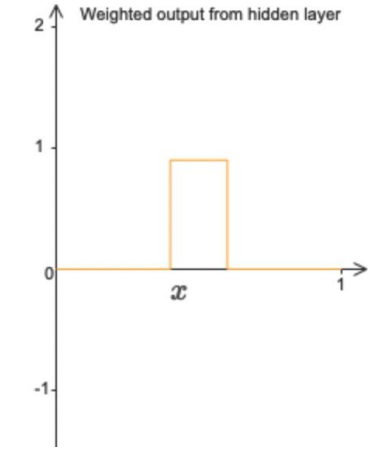
w1 是负数

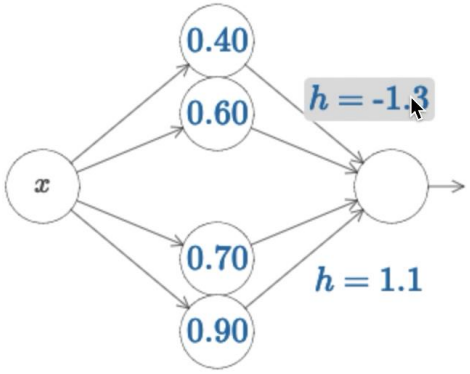


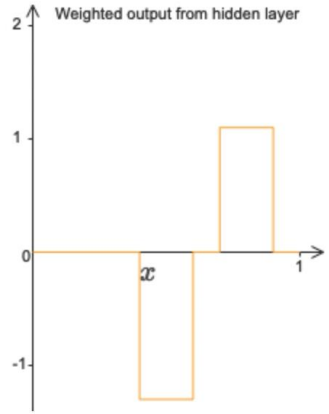
固定权重比，生成脉冲函数

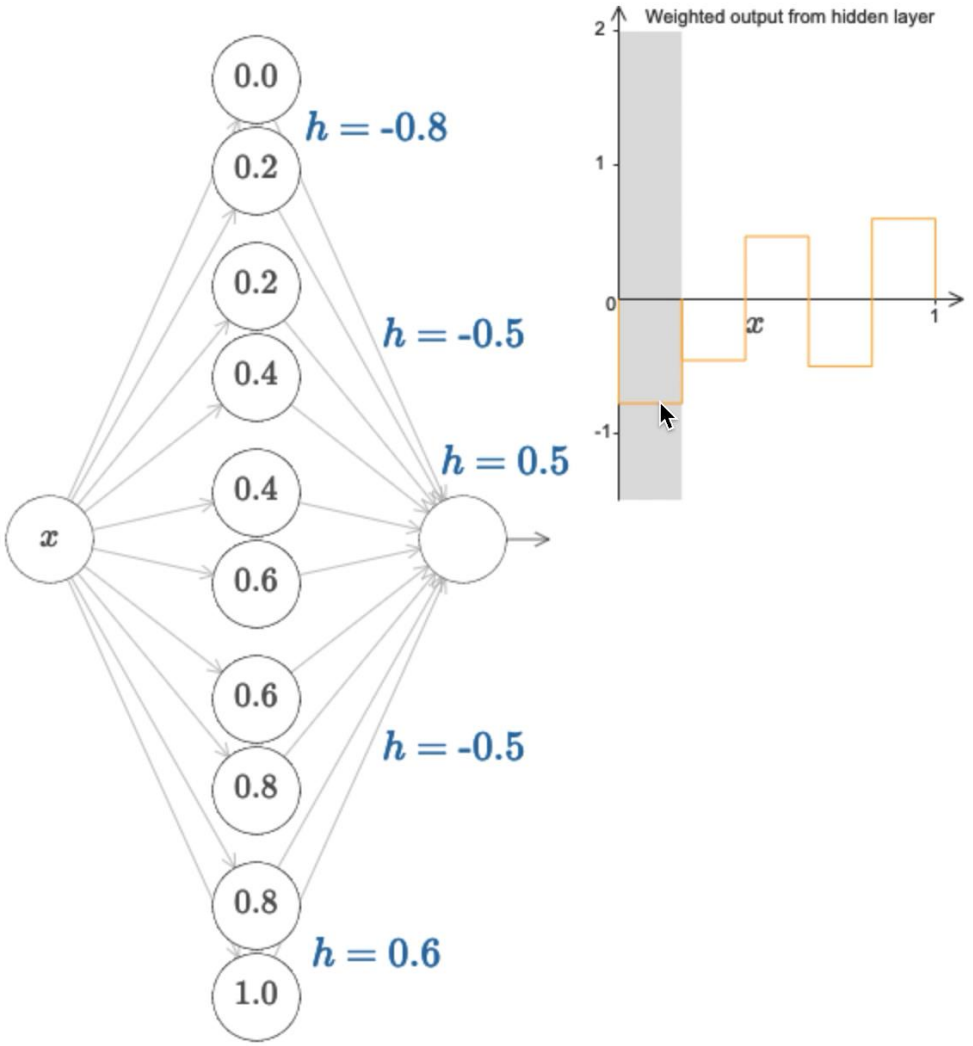


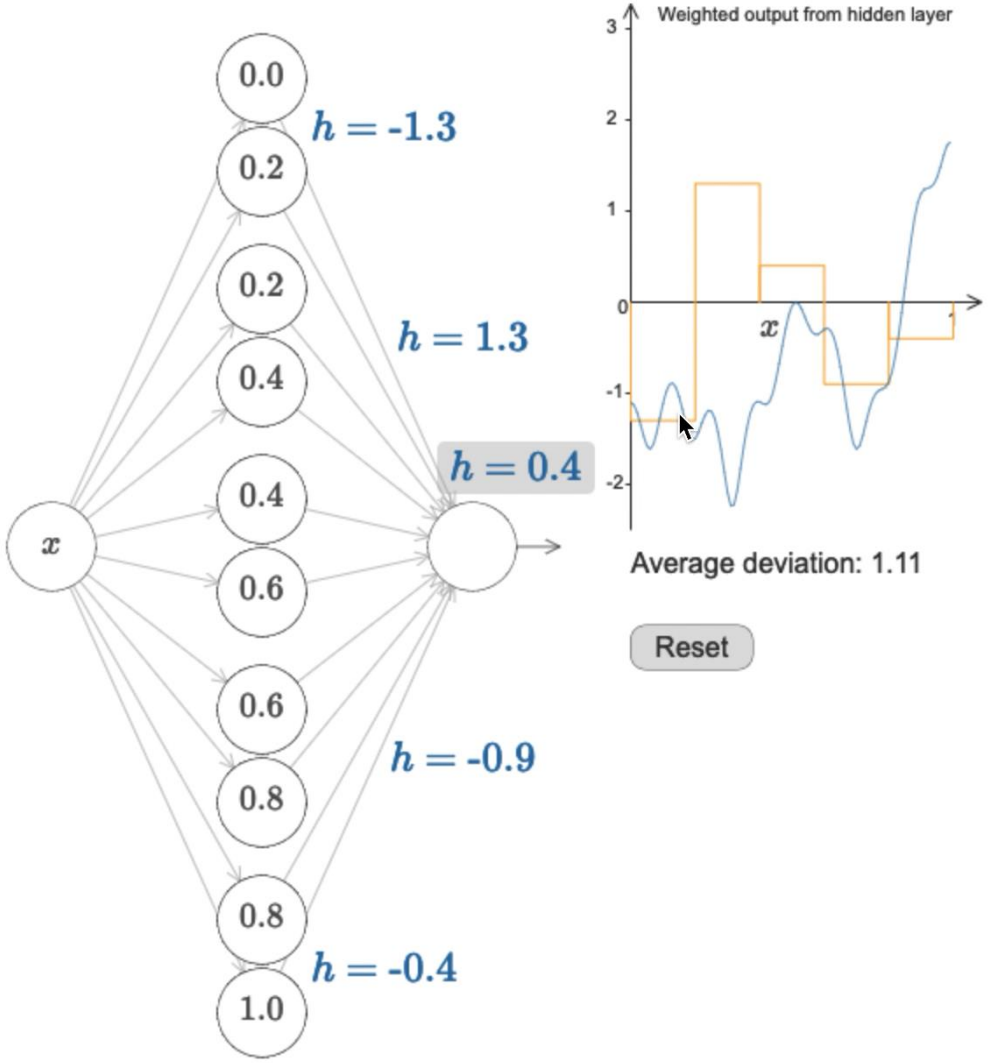






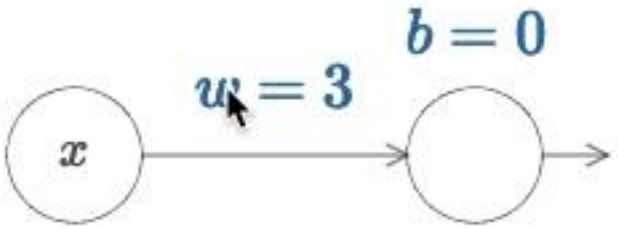


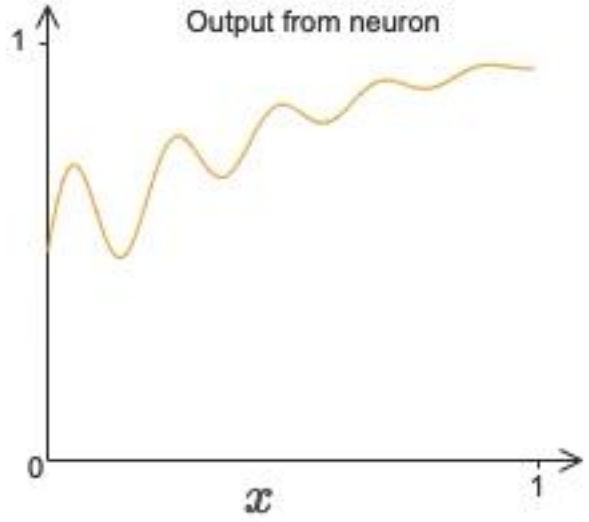

0.0
0.2
0.2
0.4
0.4
0.6
0.6
0.8
0.8
1.0
x
h = -0.8
h = -0.5
h = 0.5
h = -0.5
h = 0.6
Weighted output from hidden layer
2
1
0
-1
x
1


0.0
0.2
0.2
0.4
0.4
0.6
0.6
0.8
0.8
1.0
x
h = -1.3
h = 1.3
h = 0.4
h = -0.9
h = -0.4
Weighted output from hidden layer
3
2
1
0
-1
-2
x
Average deviation: 1.11
Reset

任意激活函数都可以变为阶跃函数

- 权重足够大，则任意函数的输出在 x 轴会被压缩到很小的范围。





深度神经网络可以拟合任意函数

- 如何让深度神经网络拟合一个特定特性的函数呢?
  - 输入图像，输出标签
  - 输入声音，输出翻译
  - 输入比赛状态，输出下一步怎么走

- 深度神经网络就是一个函数
  - 网络的参数是，权重和偏置
  - 学习网络指的是：使网络计算得到想要的函数的权重和偏置
  - 最终目的是对某函数进行建模

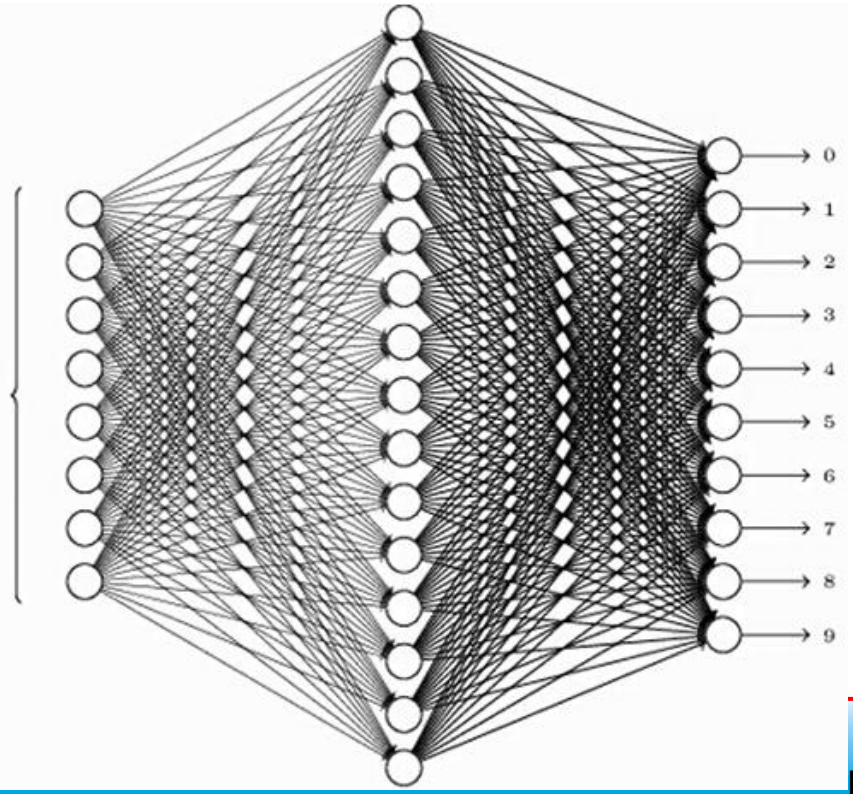
神经网络的训练 — MNIST 数据集

- MNIST
  - 250 个不同人的手写数字
  - 60000个训练集，10000个测试集
  - 每张图片由 28 x 28 个像素点组成

网络设计

- 输入层: x
  - 784个神经元(28x28)
- 隐藏层
  - 可以灵活设计
- 输出层: y(x)
  - 10个神经元

28X28

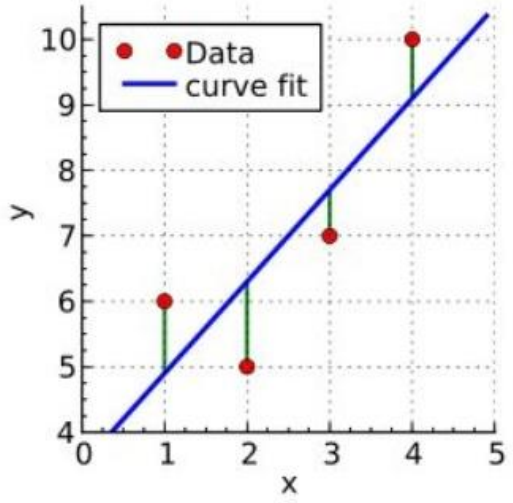


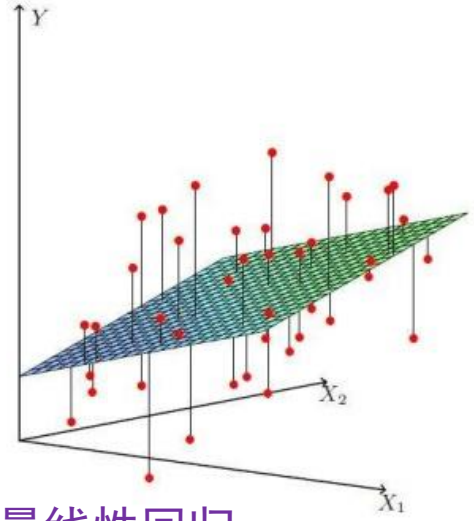
真实输出y(x)：
[0,0,0,0,0,1,0,0,0,0]

10个预测标签

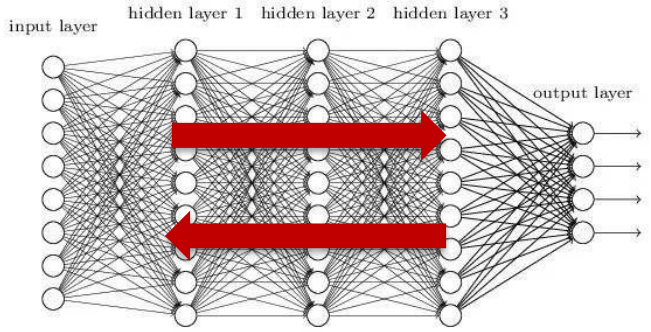
网络预测输出：
[? , ? , ? , ? , ? ,
? , ? , ? , ? , ?]

如何判断函数拟合效果?





一元变量二元变量线性回归



5？

6？

根据误差调整模型

Mean Absolute Error：平均绝对误差

$$MAE(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error：均方误差

$$MSE(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error：均方根误差

$$RMSE(y,\hat{y}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$