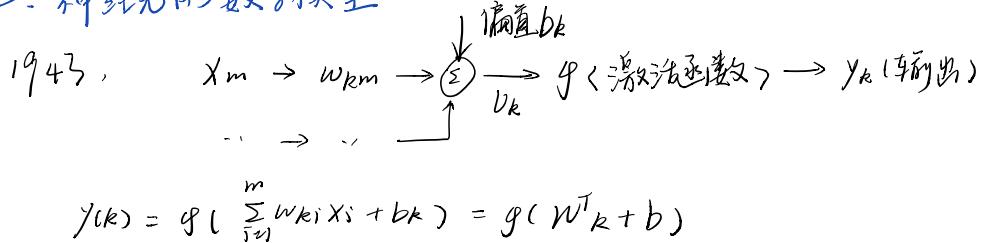


一、神经元的数学模型



二、感知器算法 (sum 几十年前的版本) Perceptron Algorithm

1957. S1. 随机选择 W, b ;

S2. 取一个训练样本 (x, y)

1> 若 $W^T x + b > 0$ 且 $y = -1$, 则: $W_{\text{新}} = W - X, b = b - 1$;

应该是 $y=+1$

$$\begin{aligned} W^T x + b &= (W - X)^T x + (b - 1) \\ &= W^T x + b - (1 \|x\|^2 + 1) \end{aligned}$$

2> 若 $W^T x + b < 0$ 且 $y = +1$, 则: $W_{\text{新}} = W + X, b = b + 1$

向正方向拉了至少 1.

S3. 再取另一个 (x, y) 回到 S2.

S4. 终止条件: 直到所有输入、输出都不满足 S2 中的条件之一.

理论上可证: 最终会分开的, 不会一直循环.

• Perceptron Algorithm 的数学证明

定义一个增广向量 \vec{x} : a. 若 $y=1$, $\vec{x} = [x]$; b. 若 $y=-1$, $\vec{x} = [x] \downarrow$

... 增广 W 如下: $W = [w]$

目的: 让 S2 判断简单.

输入 \vec{x}_i

\therefore S1. 随机选择 W ;

S2. 取一个训练样本 \vec{x}_i :

若 $W^T \vec{x}_i < 0$, 则 $W = W + \vec{x}_i$.

S3. 再取另一个 \vec{x}_i 回到 S2.

54. 终止条件：直到 all 输入、出都不满足 S2 中的条件之一。

感知器算法的收敛定理：

输入 $\{\vec{x}_i\}_{i=1 \sim N}$, 若线性可分, 即 $\exists w_{opt}$ 使:

$$w_{opt}^T \vec{x}_i > 0 \quad (i=1 \sim N)$$

则利用上述感知器算法, 经过有限步后, 得到一个 w , 使

$$w^T \vec{x}_i > 0 \quad (\sim)。$$

证明: 不失一般性, 设 $\|w_{opt}\|=1$ ($\because w_{opt}$ 与 αw_{opt} 为同一平面).

设第 k 步 w 是 $w(k)$, 且有一个 \vec{x}_i 使: $w(k)^T \vec{x}_i < 0$ (下面证明实现前提)

$$\therefore w(k+1) = w(k) + \vec{x}_i \quad (S2 \text{ 中})$$

$$\begin{aligned} \Rightarrow \|w(k+1) - \alpha w_{opt}\|^2 &= \|w(k) + \vec{x}_i - \alpha w_{opt}\|^2 \\ &= \|(\vec{w}(k) - \alpha \vec{w}_{opt}) + \vec{x}_i\|^2 \\ &= \|\vec{w}(k) - \alpha \vec{w}_{opt}\|^2 + \|\vec{x}_i\|^2 + \underbrace{2\vec{w}(k)^T \vec{x}_i}_{< 0} - \underbrace{2\alpha \vec{w}_{opt}^T \vec{x}_i}_{> 0} \end{aligned}$$

\therefore 一定可以取足够大 α , 使 $\|w(k+1) - \alpha w_{opt}\|^2 < \|w(k) - \alpha w_{opt}\|^2$

(每一项都变小, 从性质上有那趋势了)

定义: $\beta = \max_{i=1 \sim N} (\|\vec{x}_i\|)$, $\gamma = \min_{i=1 \sim N} (w_{opt}^T \vec{x}_i)$

取 $\alpha = \frac{\beta^2 + 1}{2\gamma}$, 则: $\|w(k+1) - \alpha w_{opt}\|^2 < \|w(k) - \alpha w_{opt}\|^2$

取 $D = \|w(0) - \alpha w_{opt}\|$, 则经过至多 D^2 步, w 将收敛至 αw_{opt} .

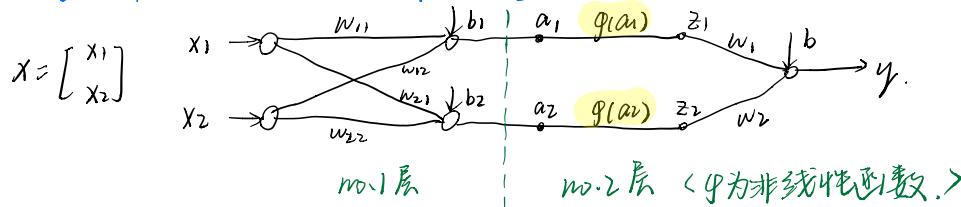
三. 人工智能的第一次寒冬

Minsky 创造了线性可分与不可分。 1969 «Perceptron»

⇒ 日常生活中很多问题是非线性可分的。 <感知器没用>

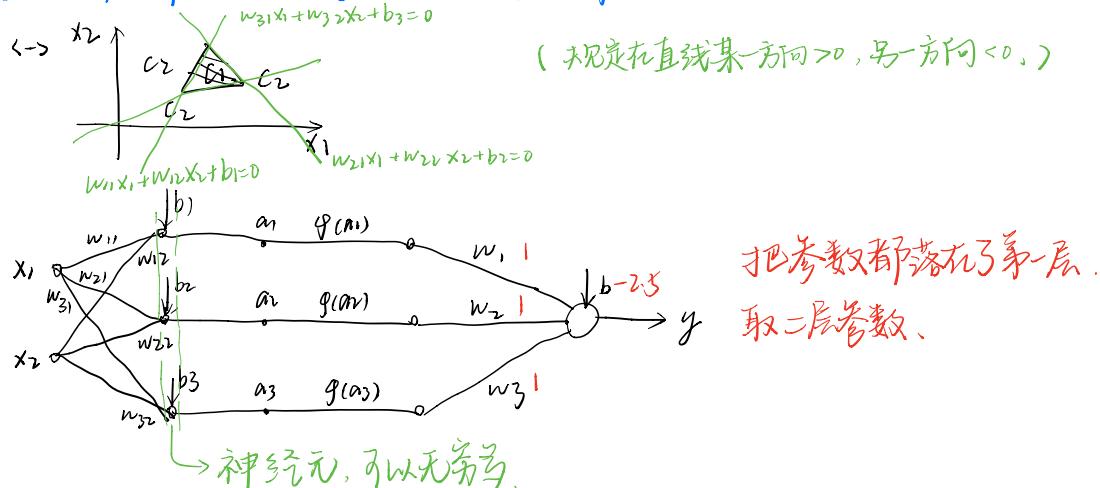
∴ 70~80年代，寒冬

四. 多层神经网络 (Multiple layer Neural Networks)

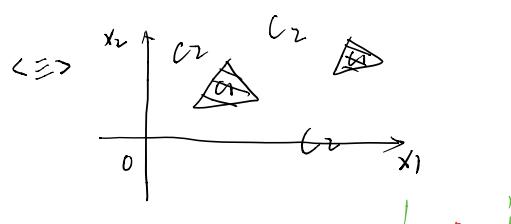


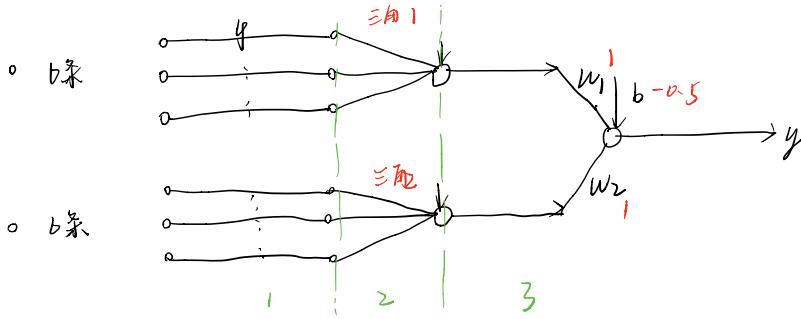
∴ 非线性函数 $g(\cdot)$ 为阶跃函数, 可处理 all 非线性问题。

五. 三层神经网络可以模拟 all 决策面.



<=> 圆 / 不规则图形 <梯度无法>





六、向后传播算法 (BP Back Propagation)

\leftrightarrow (梯度下降法求局部极值 Gradient Descent Method)

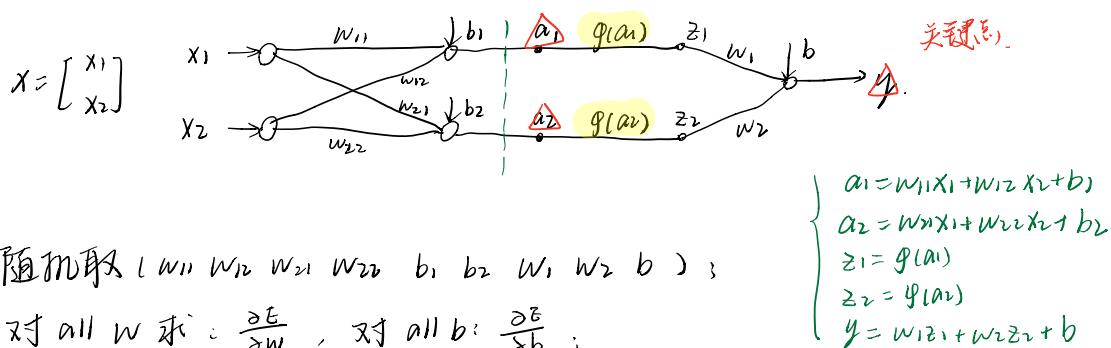
S1. 找 w_0

S2. 设 $k=0$, 假设 $\frac{df(w)}{dw}|_{w_k} = 0$, 退出

否则: $w_{k+1} = w_k - \alpha \frac{df(w)}{dw}|_{w_k}$ <用泰勒展开推导>

注:有很多变种, $\frac{df(w)}{dw}$ 只决定了一个方向.

\leftrightarrow BP: 对于输入 (x, y) , $E = \frac{1}{2} (y - \hat{y})^2$ < 目的为让 y 与 \hat{y} 一致>



① 随机取 $(w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2, w_5, w_5, b)$;

② 对 all w 求: $\frac{\partial E}{\partial w}$, 对 all b : $\frac{\partial E}{\partial b}$;

③ $w_{\text{新}} = w^{(t)} - \alpha \frac{\partial E}{\partial w}|_{w^{(t)}}$

$b_{\text{新}} = b^{(t)} - \alpha \frac{\partial E}{\partial b}|_{b^{(t)}}$

④ 当 all $\frac{\partial E}{\partial w}, \frac{\partial E}{\partial b}$ 都为 0 时, 退出循环.

$$\therefore \frac{\partial E}{\partial y} = \frac{\partial E}{\partial w} = (y - \hat{y})$$

| 先理解, 后操作.

$$\begin{aligned}
 \frac{\partial E}{\partial a_1} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z_1} \cdot \frac{\partial z_1}{\partial a_1} = (y - Y) w_1 \cdot g'(a_1) \\
 \frac{\partial E}{\partial a_2} &= (y - Y) w_2 \cdot g'(a_2)
 \end{aligned}$$

大误差时 \downarrow

$$\begin{aligned}
 \therefore \frac{\partial E}{\partial b} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial b} = (y - Y) - 1 \\
 \frac{\partial E}{\partial w_1} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial w_1} = (y - Y) z_1
 \end{aligned}$$

局部 \downarrow

$$\begin{aligned}
 \frac{\partial E}{\partial w_2} &= (y - Y) z_2 \\
 \therefore \frac{\partial E}{\partial w_{11}} &= \frac{\partial E}{\partial a_1} \cdot \frac{\partial a_1}{\partial w_{11}} = (y - Y) w_1 g'(a_1) x_1 \\
 \frac{\partial E}{\partial w_{12}} &= (y - Y) w_1 g'(a_1) x_2 \\
 \frac{\partial E}{\partial b_1} &= \frac{\partial E}{\partial a_1} \cdot \frac{\partial a_1}{\partial b_1} = (y - Y) w_1 g'(a_1) x_1 \\
 \frac{\partial E}{\partial b_2} &= (y - Y) w_2 g'(a_2) x_2
 \end{aligned}$$

再局部 \downarrow

○ 讨论 $\varphi(x)$ 、所选取(梯度) $\varphi'(x) \equiv 0$ < x >

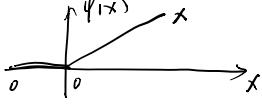
①  (sigmoid) $\varphi(x) = \frac{1}{1+e^{-x}}$
 $\varphi'(x) = \varphi(x)[1-\varphi(x)]$

(也能模拟人化决策界面)

② 
 $\varphi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
 $\varphi'(x) = 1 - [\varphi(x)]^2$

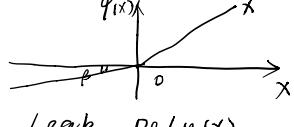
< 用以上2个变种, ∵ $\varphi'(x)$ 很简单. >

< 以下, 深度学习出来后用的 >

③ 
 $\varphi(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} = \max(0, x)$
 $\text{ReLU}(x)$
Rectified linear units

$$\varphi'(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

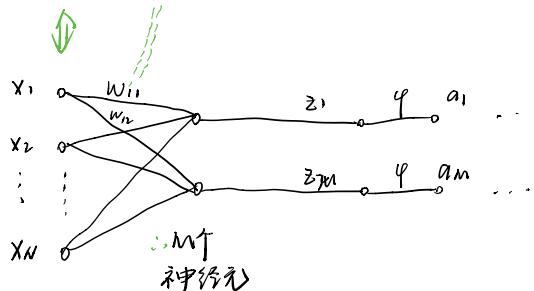
【大于0而不变了, 压缩小于0的数】

④ 
Leaky ReLU(x)
 $\varphi(x) = \begin{cases} x, & x > 0 \\ \beta x, & x \leq 0 \end{cases}$

$$\varphi'(x) = \begin{cases} 1 & , x > 0 \\ \beta & , x \leq 0 \end{cases}$$

(三) BP General 过程.

$$(N \times 1) \text{ 向量} \quad X^1 \stackrel{= a^{(1)}}{\Rightarrow} W^{(1)} \stackrel{= b^{(1)}}{\leftarrow} + b^{(1)} \quad = \quad Z^{(1)} \quad \xrightarrow{q} \quad a^{(1)} = q[Z^{(1)}] \rightarrow Z^{(2)} = W^{(2)}a^{(1)} + b^{(2)} \quad \downarrow q$$



$$\begin{aligned} \alpha^{(2)} &= \varphi[z^{(1)}] \\ &\downarrow \\ z^{(3)} &= W^{(3)}\alpha^{(2)} + b^{(3)} \\ &\downarrow \\ \langle \text{网络层 } l \rangle & \\ y = \alpha^{(l)} &= \varphi[z^{(l)}] \quad \leftarrow \quad z^{(l)} = W^{(l)}\alpha^{(l-1)} + b^{(l)} \end{aligned}$$

∴ BP 算法 (yet 只用 4 步没法) 太基本了

51. 隨機初始化 (w, b)

（前向传播）

52. 训练样本 $(x, y) < y \text{ 维度一致} \rangle$. 代入网络, 可求出所有 (x, a, y)

53. 遊式求偏导 $(\frac{\partial z}{\partial w}, \frac{\partial z}{\partial b})$

$$E = \frac{1}{2} \|y - T\|^2. \quad (\text{最简单形式, 以后更复杂的})$$

< 还是求 w 、 b ，不过每层都有 w 、 b 了 >

〈偏导计算过程〉

$$\text{设 } \delta_{(i)}^{(m)} = \frac{\partial E}{\partial z_i^{(m)}}$$

$$\textcircled{1} \quad \delta_{(i)}^{(e)} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_i^{(m)}} = (y_i - \hat{y}_i) \varphi' [z_i^{(e)}]$$

$$\textcircled{2} \quad \delta_{ij}^{(lm)} = \frac{\partial E}{\partial a_j^{(lm)}} - \frac{\partial a_i^{(lm)}}{\partial z_j^{(lm)}} = \varphi [z_i^{(lm)}] \cdot \sum_{j=1}^{S_{m+1}} \delta_j^{(m+1)} \cdot w_{ji}, \quad \langle \text{行数}, S_{m+1} \text{ 列 } \rangle$$

$1 \leq m \leq (l-1)$

$$\left\{ \begin{array}{l} \frac{\delta E}{\delta W_{ij}^{(m)}} = \delta_j^{(m)} a_i^{(m+1)} \\ \frac{\delta E}{\delta b_i^{(m)}} = \delta_i^{(m)} \end{array} \right.$$

<与之前的面貌似不同>

54. 更新式: $\begin{cases} w^{(t+1)} = w^{(t)} - \alpha \frac{\partial E}{\partial w} |_{w^{(t)}} \\ b^{(t+1)} = b^{(t)} - \alpha \frac{\partial E}{\partial b} |_{b^{(t)}} \end{cases}$