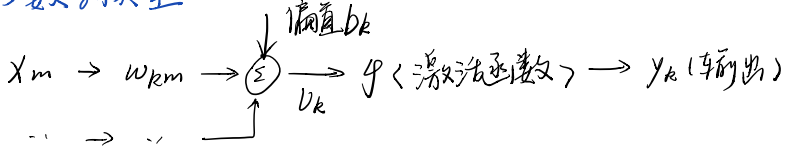
一. 神经元的数学模型

1943, 

$$y(k) = \varphi\left(\sum_{i=1}^{m} w_{ki} x_i + b_k\right) = \varphi(W^T k + b)$$

二. 感知器算法 (sum九十年前的版本) Perceptron Algorithm

1957. S1. 随机选择 W、b;

S2. 取一个训练样本 (X, y)

1) 若 $W^TX+b>0$ 且 $y=-1$, 则: $W_{新}=W-X$, $b=b-1$;

应该是 $y=+1$

$W^TX+b = (W-X)^TX+(b-1)$

$= W^TX+b-(\|X\|^2+1)$

向负方向拉了至少1 ←

2) 若 $W^TX+b<0$ 且 $y=+1$, 则: $W_{新}=W+X$, $b=b+1$

↓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all输入、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 $\tilde{x}$: a. 若 $y=1$, $\tilde{x}=\begin{bmatrix} x \\ 1 \end{bmatrix}$; b. 若 $y=-1$, $\tilde{x}=\begin{bmatrix} -x \\ -1 \end{bmatrix}$

↓ 目的: 让S2判断简单.

∵ 增广 w 如下, $W=\begin{bmatrix} w \\ b \end{bmatrix}$

输入 $\tilde{x}_i$

∴ S1. 随机选择 W;

S2. 取一个训练样本 $\tilde{x}_i$:

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

S3. 再取另一个 $\tilde{x}_i$ 回到 S2.

S4、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：（21'40''）

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

$$W^T \vec{x}_i > 0 \quad (i\sim)。$$

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

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

$\therefore W(k+1) = W(k) + \vec{x}_i$ <S2中>

$$\Rightarrow \|W(k+1) - aW_{opt}\|^2 = \|W(k) + \vec{x}_i - aW_{opt}\|^2$$

$$= \|(W(k) - aW_{opt}) + \vec{x}_i\|^2$$

$$= \|W(k) - aW_{opt}\|^2 + \|\vec{x}_i\|^2 + \underbrace{2W(k)^T\vec{x}_i}_{<0} - \underbrace{2aW_{opt}\vec{x}_i}_{>0}$$

$\therefore$一定可取足够大a，使$\|W(k+1) - aW_{opt}\|^2 < \|W(k) - aW_{opt}\|^2$

<每一项都变小，从定性上有那趋势了>

定义：$\beta = \max\limits_{i=1\sim N}(\|\vec{x}_i\|)$，$\gamma = \min\limits_{i=1\sim N}(W_{opt}^T x_i)$

取$a = \frac{\beta^2+1}{2\gamma}$，则$\|W(k+1) - aW_{opt}\|^2 < \|W(k) - aW_{opt}\|^2 - 1$

取$D = \|W(0) - aW_{opt}\|$，则经过至多D^2步，W将收敛至aW_{opt}。

三、人工智能的第一次寒冬

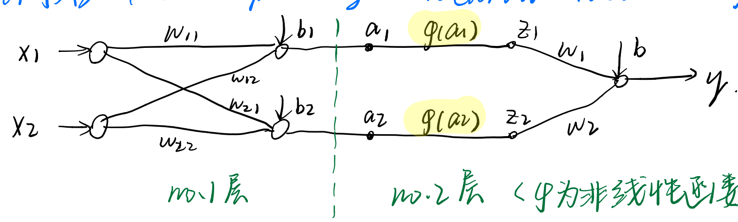
Minsky 创造了线性可分与⋯不可分. 1969《Perceptron》

=> 日常生活中很多问题是非线性可分的. <感知器设法用>

∴ 70~80年代, 寒冬

四、多层神经网络 (Multiple layer Neurod Networks)

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

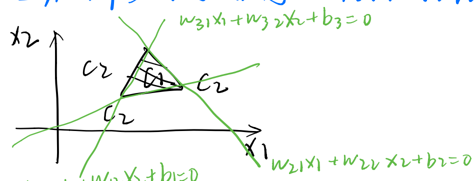


若无φ( ), y和x仍为线性关系, 与单一神经元无区别.

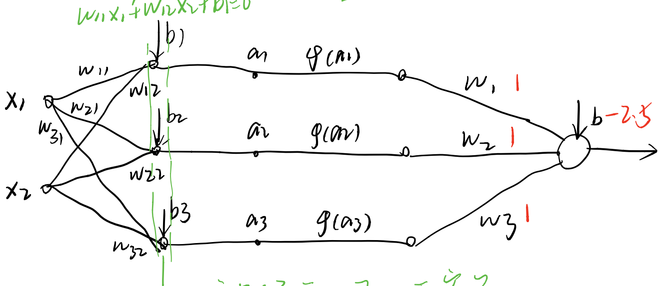
∴ 非线性函数 φ( ) 为阶跃函数, 可处理all非线性问题.

五、三层神经网络可以模拟all决策面.

<一>



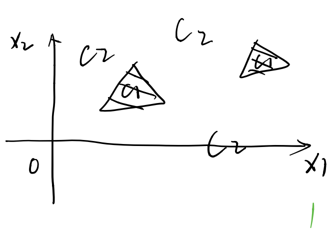
(规定在直线某一方向>0, 另一方向<0.)

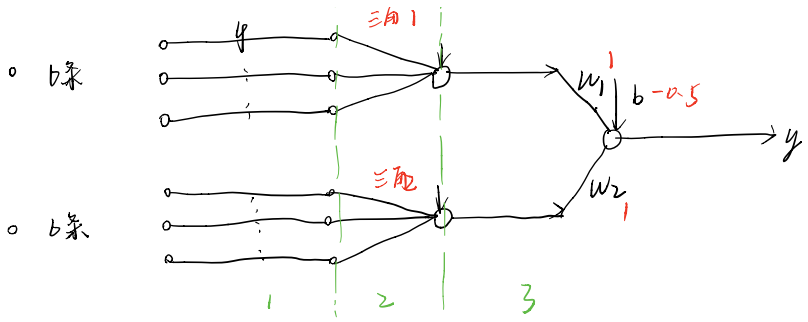


把参数都落在了第一层. 取二层参数.

<二> 圆 / 不规则图形 <微积元法>

<三>





六、向后传播算法（BP Back Propogation）

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

S1. 找一 w_0

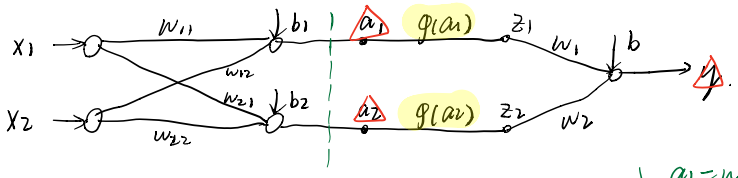
S2. 设 $k=0$，假设 $\left.\frac{df(w)}{dw}\right|_{w_k}=0$，退出

否则：$w_{k+1}=w_k-\alpha\left.\frac{df(w)}{dw}\right|_{w_k}$ <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。

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

$x=\begin{bmatrix}x_1\\x_2\end{bmatrix}$



$$\begin{cases}a_1=w_{11}x_1+w_{12}x_2+b_1\\a_2=w_{21}x_1+w_{22}x_2+b_2\\z_1=g(a_1)\\z_2=g(a_2)\\y=w_1z_1+w_2z_2+b\end{cases}$$

① 随机取 $(w_{11}\ w_{12}\ w_{21}\ w_{22}\ b_1\ b_2\ w_1\ w_2\ b)$；

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

③ $w_{新}=w_{旧}-\alpha\left.\frac{\partial E}{\partial w}\right|_{w_{旧}}$

$b_{新}=b_{旧}-\alpha\left.\frac{\partial E}{\partial b}\right|_{b_{旧}}$

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

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

关键点，名称点。

$$\frac{\partial E}{\partial a_1} = \frac{dE}{dy} \cdot \frac{\partial y}{\partial z_1} \cdot \frac{dz_1}{da_1} = (y-Y)w_1 \cdot \varphi'(a_1)$$

$$\frac{\partial E}{\partial a_2} = (y-Y)w_2\varphi'(a_2)$$

关键的〈逆推多〉

↓

局部

↓

再局部

$\because$ $$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial b} = (y-Y)\cdot 1$$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial w_1} = (y-Y)z_1$$

$$\frac{\partial E}{\partial w_2} = (y-Y)z_2$$

$\therefore$ $$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial a_1} \cdot \frac{\partial a_1}{\partial w_{11}} = (y-Y)w_1\varphi'(a_1)x_1$$

$$\frac{\partial E}{\partial w_{12}} = (y-Y)w_1\varphi'(a_1)x_2$$

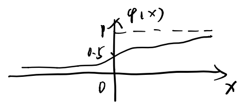
$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial a_1} \cdot \frac{\partial a_1}{\partial b_1} = (y-Y)w_1\varphi'(a_1)\times 1$$

$$\frac{\partial E}{\partial w_{21}} = (y-Y)w_2\varphi'(a_2)x_{z1}$$

$$\frac{\partial E}{\partial w_{22}} = (y-Y)w_2\varphi'(a_2)x_2$$

$$\frac{\partial E}{\partial b_2} = (y-Y)w_2\varphi'(a_2)$$

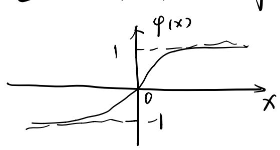
○ 讨论 $\varphi(\cdot)$、 阶跃(除0点) $\varphi'(x) \equiv 0$ 〈X〉

① 

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

$\varphi'(x) = \varphi(x)[1-\varphi(x)]$

（也能模拟任意的决策面）

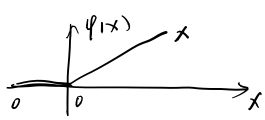
② 

$\varphi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

$\varphi'(x) = 1 - [\varphi(x)]^2$

〈用以上2个变种，$\because$ $\varphi'(x)$很简单.〉

〈 以下，深度学习出来后用的 〉

③ 

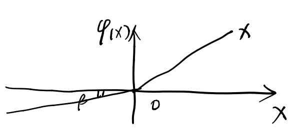
ReLu(x)

Rectified Linear units

$$\varphi(x) = \begin{cases} 0, & x \le 0 \\ x, & x > 0 \end{cases} = \max(0, x)$$

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

[大于0的不管了，压缩小于0的数.]

④ 

Leak ReLu(x)

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

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

<三> BP General 过程.

$x \overset{=a^{(0)}}{\Rightarrow} W^{(1)}x^{a^{(0)}}+b^{(1)} = z^{(1)} \xrightarrow{\varphi} a^{(1)}=\varphi[z^{(1)}] \longrightarrow z^{(2)}=W^{(2)}a^{(1)}+b^{(2)}$

(N×1 向量) M×N N×1 M×1 M×1

$\downarrow\varphi$

$a^{(2)}=\varphi[z^{(2)}]$

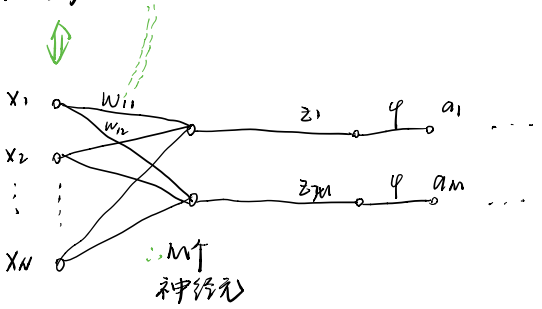
$\downarrow$

$z^{(3)}=W^{(3)}a^{(2)}+b^{(3)}$

$\downarrow$

<网络层layer>

$y=a^{(l)}=\varphi[z^{(l)}] \Leftarrow z^{(l)}=W^{(l)}a^{(l-1)}+b^{(l)}$



∴, BP算法 <Yet只用4步设法> 太基本了

S1. 随机初始化 (W, b)

S2. 训练样本 (X, Y) <Y、y维度一致>. 代入网络. 可求出所有 (z, a, y)

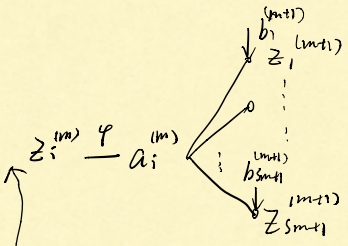
<前向传播>

S3. 链式求偏导 $\left(\frac{\partial E}{\partial W}, \frac{\partial E}{\partial b}\right)$

$E=\frac{1}{2}\|y-Y\|^2$. (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>



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

① $\delta_{(i)}^{(l)}=\frac{\partial E}{\partial y_i}\cdot\frac{\partial y_i}{\partial z_i^{(m)}}=(y_i-Y_i)\varphi'[z^{(l)}]$

② $\delta_{(i)}^{(m)}=\frac{\partial E}{\partial a_i^{(m)}}\cdot\frac{\partial a_i^{(m)}}{\partial z_i^{(m)}}=\varphi[z_i^{(m)}]'\cdot\sum_{j=1}^{(m+1)}\delta_j^{(m+1)}\cdot w_{ji}$. <j层数, i层上的第i个>

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

$$\begin{cases}\frac{\partial E}{\partial W_{ij}^{(m)}}=\delta_j^{(m)}a_i^{(m-1)}\\ \frac{\partial E}{\partial b_i^{(m)}}=\delta_i^{(m)}\end{cases}$$

<与之前的i、j视图以不同>.

S4. 更新：$\begin{cases} w^{新} = w^{旧} - \alpha \frac{\partial L}{\partial w}\Big|_{w^{旧}} \\ b^{新} = b^{旧} - \alpha \frac{\partial L}{\partial b}\Big|_{b^{旧}} \end{cases}$