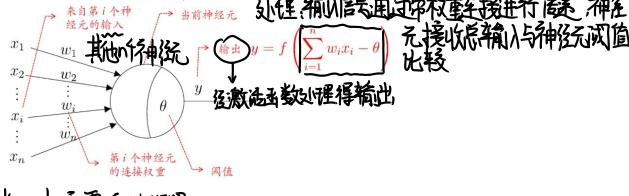


## 数学上用函数表示

智能体认知  $\Rightarrow$  寻找事物间联系来认识世界 在大脑中不断形成各种函数

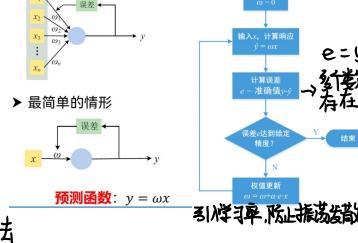
神经元模型：接收处理、传递信息

第一个神经元数学模型：M-P模型



米 W 权重需手动设置

Rosenblatt 感知机模型与学习规则：有了自己调整参数的能力



梯度下降法

根据斜率来调整  $\omega$ , 找到代价函数的最小点

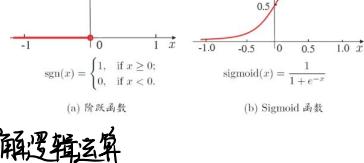


参数  $\omega$  的更新过程  $\omega_{j+1} = \omega_j - \eta \frac{de}{d\omega}$

- BGD (Batch Gradient Descent) 批量梯度下降：每一次迭代都用全部样本的损失的平均值来计算梯度，所以如果全部样本非常多，模型的收敛速度就会非常慢，训练时间非常长，因此不常用；
- SGD (Stochastic Gradient Descent) 随机梯度下降：每次迭代只用一个样本来对参数进行更新，速度快了，但是准确度下降，现在也不是很常用；
- MBGD (Mini-Batch Gradient Descent) 小批量梯度下降：现将样本分组，然后每一个组进行一次迭代，这个常用。

激活函数  $\Rightarrow$  智能体更擅长做分类而非拟合

理想激活函数为阶跃函数：0表抑制，1表激活  $\Rightarrow$  不连续不光滑，常采用 sigmoid, tanh



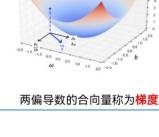
引入均方误差

$$\text{平均误差 } \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 = \frac{1}{m} (\sum_{i=1}^m x_i^2 w^2 - 2 \sum_{i=1}^m (x_i y_i) w + \sum_{i=1}^m y_i^2)$$

$$= w^2 - 2 w \sum_{i=1}^m x_i y_i + \sum_{i=1}^m y_i^2$$

$$\text{求导令为0有 } w_{\min} = -\frac{b}{2a} = \frac{\sum x_i y_i}{\sum x_i^2} \text{ 计算量太大}$$

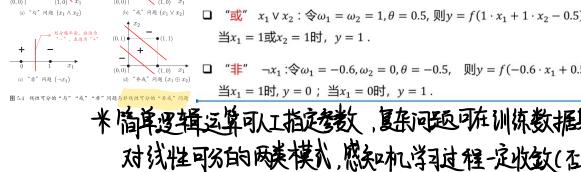
线性预测函数更一般的形式为  $y = \omega x + b$



两个偏导数的合向量称为梯度

感知机求解逻辑运算

理想激活函数为阶跃函数：0表抑制，1表激活  $\Rightarrow$  不连续不光滑，常采用 sigmoid, tanh



**图 5.3 两个输入神经元的感知机网络结构示意图**

逻辑简单运算可手工指定参数，复杂问题可在训练数据集上学习到权重  $w$  与阈值  $b$

对线性可分的两类模式，感知机学习过程一定收敛（否则振荡） 或非

仅能输出 0/1，输出层感知机学习能力非常有限，只能解决线性可分问题，无法解决 XOR 问题

对非线性可分问题可增加加隐藏层进行解决

多层感知机/多层次前馈神经网络



隐藏层与输出层都是具有激活函数的功能神经元

误差反向传播算法

给定训练数据集  $D = \{(x_i, y_i)\}, x_i \in R^d, y_i \in R^l, (i = 1, 2, \dots, m)$ ，即输入示例由  $d$  个属性描述，输出  $l$  维实值向量

● 为方便讨论，给定一个拥有  $d$  个输入神经元  $l$  个输出神经元  $q$  的多层神经元的多层前向神经网络结构

$\theta_j$ : 第  $j$  层第  $j$  个神经元阈值

$v_{ih}$ : 隐含层第  $i$  个神经元阈值

$w_{ij}$ : 输入层与隐含层神经元之间的连接权重

$w_{ij}$ : 隐含层与输出层神经元之间的连接权重

对于样例  $(x_i, y_i)$ ，假设网络的实际输出为  $\hat{y}_i$

□ 前向计算 step1:  $\hat{y}_i = f(a_{i1} - \gamma_{i1})$ ,  $a_{i1} = \sum_{h=1}^H v_{ih} x_i$

step2:  $\hat{y}_i = f(\beta_j - \theta_j)$ ,  $\beta_j = \sum_{h=1}^H w_{jh} \hat{y}_i$

step3:  $E_{ik} = \frac{1}{2} \sum_{j=1}^l (\hat{y}_{ij} - y_{ij})^2$

□ 反向传播

权重： $v_{ih}, w_{ij}$  调整  $\theta_j, \gamma_{ih}$  ( $i = 1, \dots, d; h = 1, \dots, l; j = 1, \dots, q$ )

因此网络中需要  $(d + l + 1) \times (l + 1) \times q$  个参数需要优化

□ 参数优化：BP 是一个迭代式学习算法，在迭代的每一轮中采用广义的感知机学习规则对参数进行更新

估计任意参数  $v$  的更新公式为  $v \leftarrow v + \Delta v$

BP 学习算法

激活函数sigmoid函数： $f(x) = \frac{1}{1 + e^{-x}}$ ,  $f'(x) = f(x)(1 - f(x))$

● BP 算法是梯度下降准则，以目前的负梯度方向对参数进行调整

● 对误差  $E$ ，给定学习率  $\eta$

$\Delta \theta_k = -\eta \frac{\partial E}{\partial \theta_k}$

$\frac{\partial E_k}{\partial w_{jh}} = \frac{\partial E_k}{\partial \beta_j} \frac{\partial \beta_j}{\partial w_{jh}} \frac{\partial \beta_j}{\partial \theta_j}$  !

输出层梯度项：

$\frac{\partial E_k}{\partial \beta_j} = -\frac{\partial E_k}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial \beta_j}$

$= -(\hat{y}_j^k - y_{kj})^2 f'(\beta_j - \theta_j)$

$= \hat{y}_j^k (1 - \hat{y}_j^k) (y_{kj} - \hat{y}_j^k)$  (5.10)

$\Delta w_{jh} = \eta w_{jh} \hat{y}_j^k$  (5.11)

● 类似地可以推导出

$\Delta \theta_j = \eta \theta_j$  (5.12)

$\Delta v_{ih} = \eta v_{ih} x_i$  (5.13)

$\Delta \gamma_{ih} = \eta \gamma_{ih}$  (5.14)

其中

$\frac{\partial E_k}{\partial \theta_k} = \frac{\partial E_k}{\partial b_h}$

$= -\sum_{j=1}^l \frac{\partial E_k}{\partial \beta_j} \frac{\partial \beta_j}{\partial \theta_k}$

$= -\sum_{j=1}^l \frac{\partial E_k}{\partial \beta_j} f'(\alpha_h - \gamma_{ih})$

$= -\sum_{j=1}^l w_{jh} \beta_j f'(\alpha_h - \gamma_{ih})$

$= -\sum_{j=1}^l w_{jh} \hat{y}_j^k f'(\alpha_h - \gamma_{ih})$

$= -\sum_{j=1}^l w_{jh} \hat{y}_j^k (1 - \hat{y}_j^k) (\alpha_h - \gamma_{ih})$

$= -\sum_{j=1}^l w_{jh} \hat{y}_j^k \alpha_h (1 - \hat{y}_j^k)$

$= -\sum_{j=1}^l w_{jh} \hat{y}_j^k \alpha_h$

<