第2章 感知机

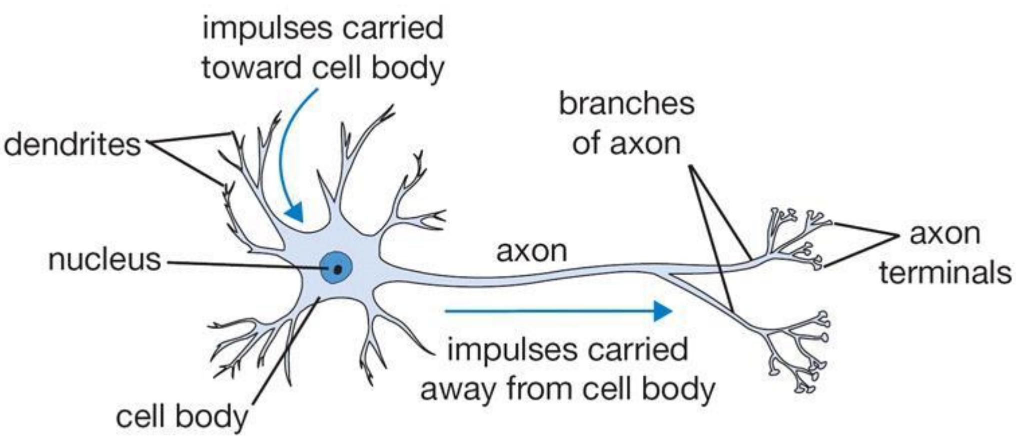
主要内容

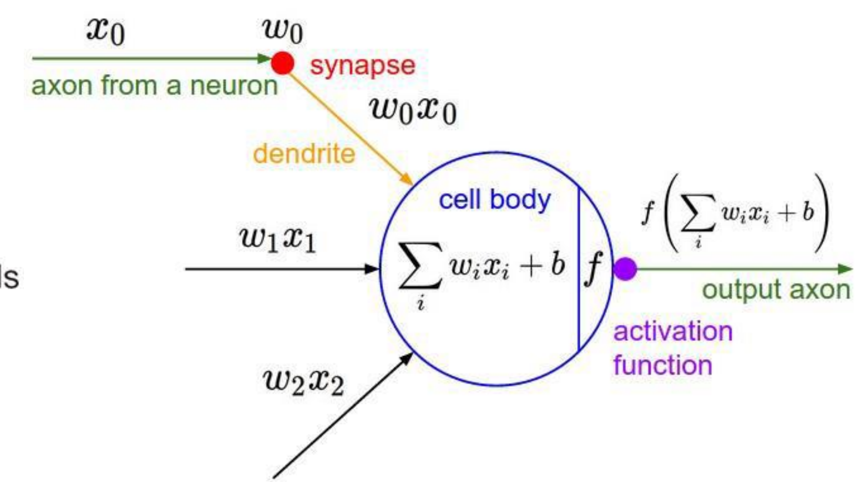
- 感知机模型
- 感知机学习策略
- 感知机学习算法
- BP算法

2.1 感知机模型

1、感知机(Perceptron)

感知机是1957年，由Rosenblatt提出，是**神经网络和支持向量机**的基础





输入→神经元→非线性激活函数→输出

激活函数

理想激活函数是阶跃函数，0表示抑制神经元而1表示激活神经元

阶跃函数具有不连续、不光滑等不好的性质，常用的是 Sigmoid 函数

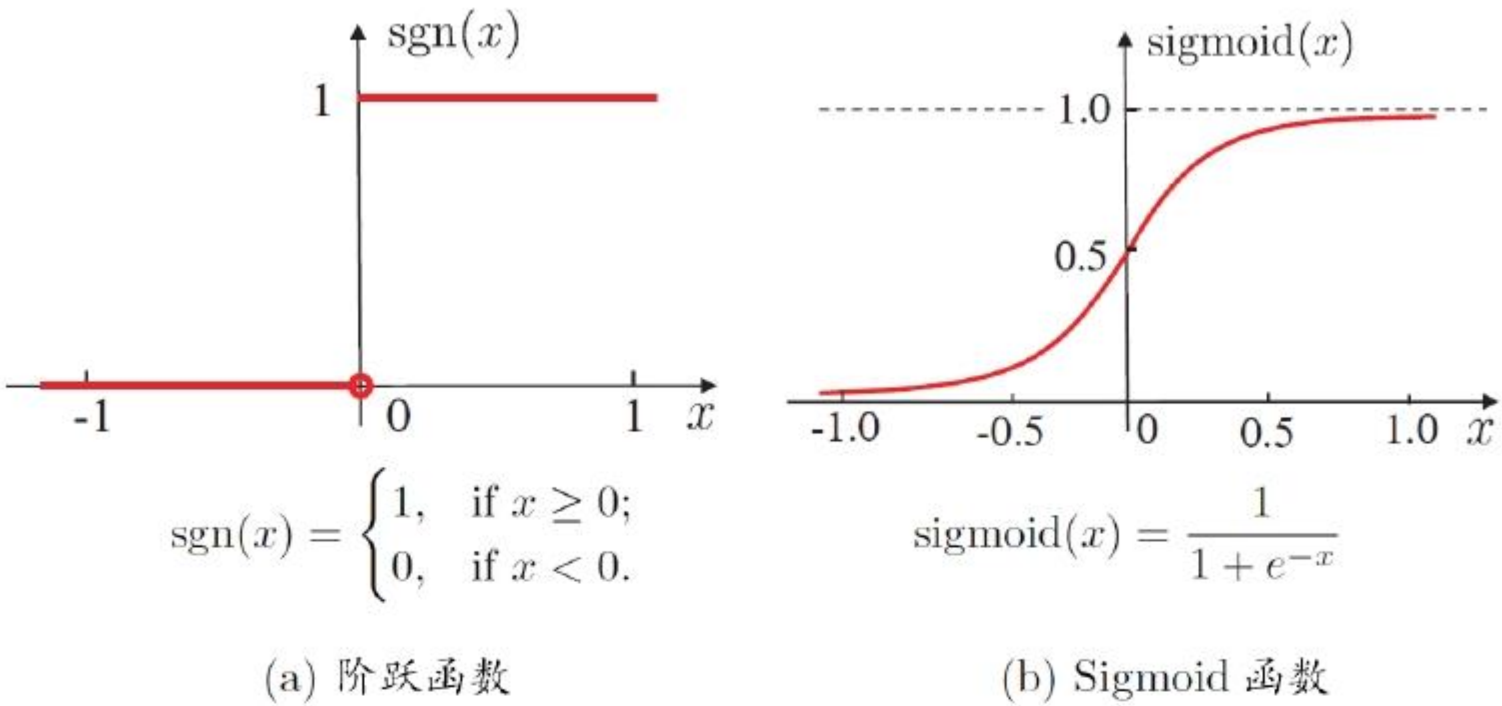


图 5.2 典型的神经元激活函数

2、定义2.1（感知机）

假设输入空间(特征空间)是$X \in R^n$，输出空间是Y={+1，-1}，输入$x \in X$表示实例的特征向量，对应于输入空间(特征空间)的点，输出$y \in Y$表示实例的类别，由输入空间到输出空间的函数

f(x)=sign(w.x+b)，称为感知机

其中w和b为感知机模型参数，$w \in R^n$为权值向量（或权值），$b \in R$为偏置。w.X表示w和x的内积，符号函数：

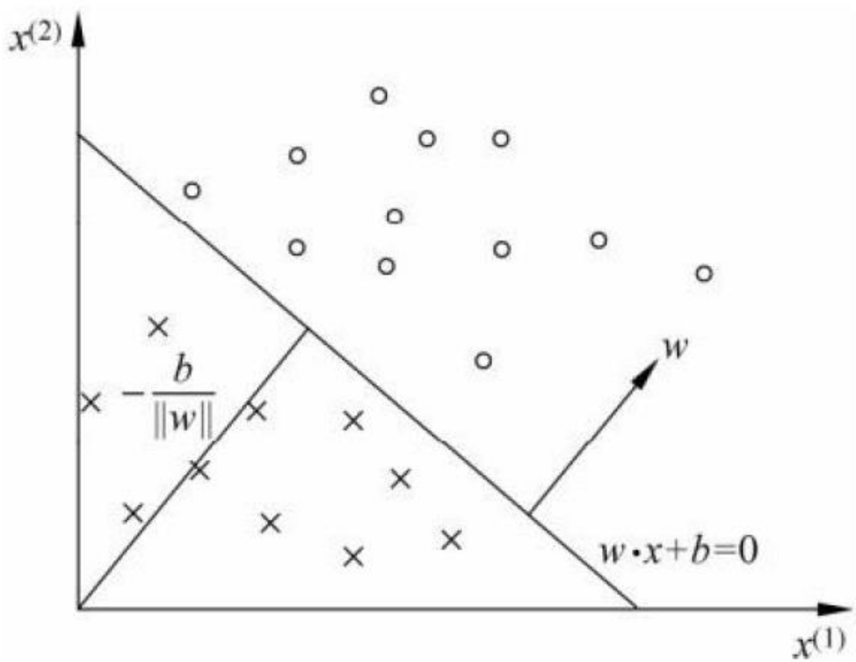
$$sign(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

感知机模型的假设空间是定义在特征空间中的所有线性分类模型(或者线性分类器)，即函数集合{f|f(x)=w.x+b}

补充：内积(点积)：$w.b=wb^T$,如w=(w1,w2,…wn)，b=(b1,b2,..bn)，则内积w.b=w1b1+w2b2+…+wnbn

3、几何解释

线性方程 w.x+b=0，对应于特征空间R^n中的一个超平面S，w法向量，b截距。



2.2 感知机感知机学习策略

1、数据集的线性可分性

定义2.2(数据集的线性可分性) 给定一个数据集

$$T=\{(x_1,y_1), (x_2,y_2),\ldots(x_N,y_N)\}$$

其中，$x_i \in X = R^n$，$y_i \in Y = \{+1,-1\}$，$i = 1,2,\ldots,N$，如果存在某个超平面S

$$w.x+b=0$$

能够将数据集的正实例点和负实例点完全正确地划分到超平面的两侧，即

对所有$y_i = +1$的实例i，有$w \cdot x_i + b > 0$，

对所有$y_i = -1$的实例i，有$w \cdot x_i + b < 0$，

则称数据集T为线性可分数据集(linearly separable data set)；否则，称数据集T线性不可分。

2、如何定义损失函数？

误分类点到超平面的总距离：

- 点(x_0,y_0)到直线$AX+BY+C=0$的距离 $d=\left|\frac{Ax_0+by_0+C}{\sqrt{A^2+B^2}}\right|$
- 输入空间R^n中任意一点x_0到超平面S的距离 $\frac{1}{\|w\|}|w.x_0+b|$
- 对于误分类点(x_i,y_i)到超平面的距离 $-\frac{1}{\|w\|}y_i(w.x_i+b)$
- 假设超平面S的误分类点集合为M，那么所有误分类点到超平面S的总距离

$$-\frac{1}{\|w\|}\sum_{x_i\in M}y_i(w.x_i+b)$$

3、损失函数

给定一个数据集T={(x_1,y_1), (x_2,y_2),...(x_N,y_N)}，其中x_i∈X = R^n，y_i∈Y ={+1,-1}， i=1,2,...,N。感知机sign(w·x+b)学习的损失函数定义为：

$$L(w,b) = -\sum_{x_i \in M} y_i (w.x_i + b)$$ M为误分类点的集合

感知机学习策略：

$$\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i (w \bullet x_i + b)$$

思考：为什么可以损失函数不考虑1/||w||？

2.3 感知机学习算法

梯度下降法

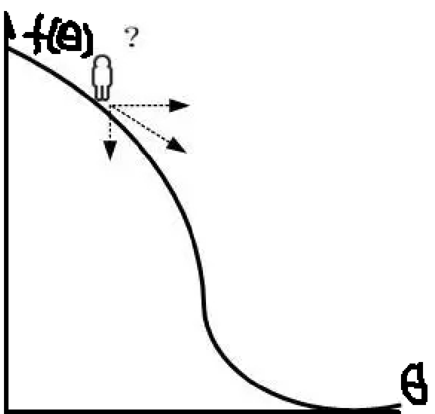
设损失函数为L(w)，用一阶泰勒展开

$$L(w) \approx L(w^{(k)}) + (w - w^{(k)})\nabla L(w)$$

$$\because \ L(w) - L(w^{(k)}) < 0 \ \therefore \ (w - w^{(k)})\nabla L(w) < 0$$

$$\Rightarrow w - w^{(k)} = -\alpha \frac{\nabla J(w)}{\left\|\nabla J(w)\right\|} = -\eta \nabla J(w)$$

$$\Rightarrow w = w^{(k)} - \alpha \frac{\partial J(w)}{\partial w}$$



1、求解最优化问题的算法

$$\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i (w \bullet x_i + b)$$

- 随机梯度下降法
- 首先任意选择一个超平面w_0,b_0，然后不断极小化目标函数（损失函数L的梯度）
- 选取误分类点更新 $\nabla_w L(w,b) = -\sum_{xi \in M} y_i x_i$

$$\nabla_b L(w,b) = -\sum_{xi \in M} y_i$$

$$w \leftarrow w + \eta y_i x_i$$

$$b \leftarrow b + \eta y_i$$

η是学习步长（学习率），取值在0和1之间

2、感知机学习算法的原始形式

算法**2.1**（感知机学习算法的原始形式）

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$,其中$x_i \in X=R^n$，$y_i \in Y=\{-1,+1\}$，$i=1,2,\ldots,N$；学习率$(0<\eta \le 1)$；

输出：w,b；感知机模型$f(x)=\mathrm{sign}(w \cdot x+b)$。

（1）选取初值w_0,b_0

（2）在训练集中选取数据$(x_i，y_i)$

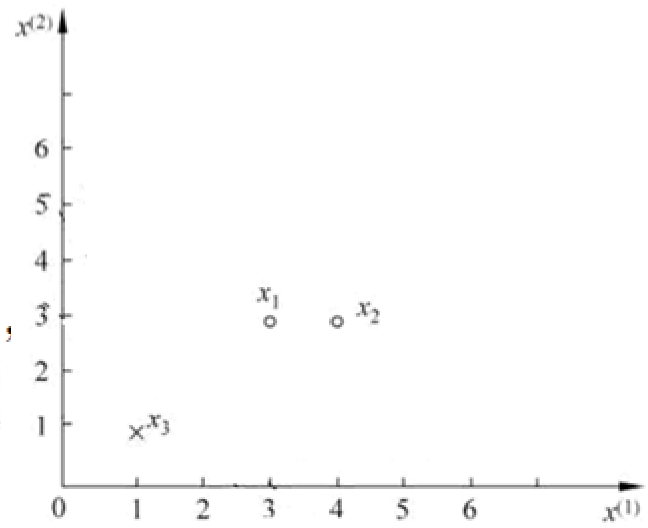
（3）如果$y_i(w \cdot x_i+b) \le 0$

$$w \leftarrow w+\eta y_i x_i$$

$$b \leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点。

例2.1 如图2.2所示的训练数据集，其正实例点是$x_1=(3,3)^T$，$x_2=(4,3)^T$，负实例点是$x_3=(1,1)^T$，试用感知机学习算法的原始形式求感知机模型$f(x)=\text{sign}(w\cdot x+b)$。这里，$w=(w^{(1)},w^{(2)})^T$，$x=(x^{(1)},x^{(2)})^T$。



解：构建最优化问题，η=1

（1）取初值$w_0=0$，$b_0=0$

（2）对$x_1=(3,3)^T$，$y_1(w_0\cdot x_1+b_0)=0$，未能被正确分类，

更新w,b　$w_1=w_0+y_1x_1=(3,3)^T$，$b_1=b_0+y_1=1$

得到线性模型

$$w_1\cdot x+b_1=3x^{(1)}+3x^{(2)}+1$$

（3）对x_1,x_2，显然，$y_i(w_1\cdot x_i+b_1)>0$，被正确分类，不修改w,b；

对$x_3=(1,1)^T$，$y_3(w_1\cdot x_3+b_1)<0$，被误分类，更新w,b。

$$w_2=w_1+y_3x_3=(2,2)^T，\quad b_2=b_1+y_3=0$$

得到线性模型

$$w_2\cdot x+b_2=2x^{(1)}+2x^{(2)}$$

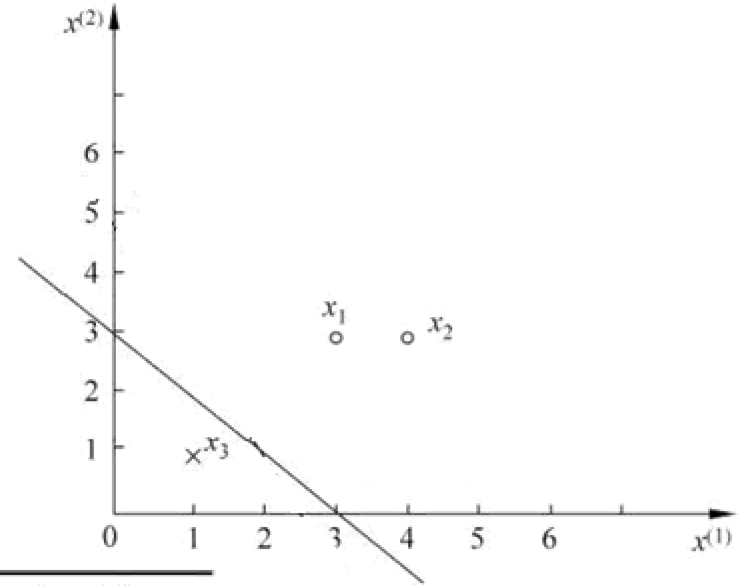
如此继续下去，直到：

$w_7=(1,1)^T, b_7=-3$

$w_7 \cdot \chi + b_7 = x^{(1)}+x^{(2)}-3$

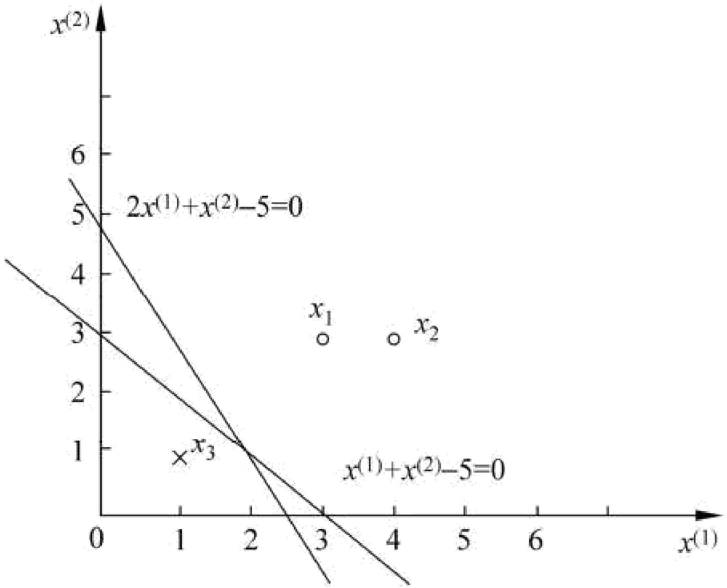
分离超平面为$x^{(1)}+x^{(2)}-3=0$

感知机模型为$f(x)=\text{sign}(x^{(1)}+x^{(2)}-3)$



迭代次数	误分类点	w	b	$w \cdot x+b$
0		0	0	0
1	x_1	$(3,3)^T$	1	$3x^{(1)}+3x^{(2)}+1$
2	x_3	$(2,2)^T$	0	$2x^{(1)}+2x^{(2)}$
3	x_3	$(1,1)^T$	−1	$x^{(1)}+x^{(2)}-1$
4	x_3	$(0,0)^T$	−2	−2
5	x_1	$(3,3)^T$	−1	$3x^{(1)}+3x^{(2)}-1$
6	x_3	$(2,2)^T$	−2	$2x^{(1)}+2x^{(2)}-2$
7	x_3	$(1,1)^T$	−3	$x^{(1)}+x^{(2)}-3$
8	0	$(1,1)^T$	−3	$x^{(1)}+x^{(2)}-3$

采用不同的初值或者选取不同的误分类点，分离超平面可以不同。



如误分类点顺序x_1，x_3， x_3， x_3， x_2， x_3， x_3， x_3， x_1， x_3， x_3.
分离超平面$2x^{(1)}+x^{(2)}-5=0$

3、感知机学习算法的对偶形式

基本想法：将w和b表示为实例x_i和标记y_i的线性组合的形式，通过求解其系数而求得w和b。

在算法2.1中，假设初始值w_0,b_0均为0。对误分类点（x_i，y_i）通过

$$w \leftarrow w + \eta y_i x_i$$

$$b \leftarrow b + \eta y_i$$

逐步修改w,b，设修改n次，则w,b关于（x_i，y_i）的增量分别是$a_i y_i x_i$和$a_i y_i$，这里$a_i = n_i \eta$。学习到的w,b可以分别表示为

$$w = \sum_{i=1}^{N} a_i y_i x_i$$

$$b = \sum_{i=1}^{N} a_i y_i$$

$a_i \geq 0$，i=0,1,..N，当η =1时，表示第i个实例更新的次数。

实例点更新次数越多，说明它更接近于超平面，越难正确分类

算法2.2 感知机学习算法的对偶形式

输入：线性可分的数据集$T=\{(x_1, y_1),(x_2, y_2),\ldots,(x_N,y_N)\}$，其中$x_i \in R^n$，$y_i \in \{-1,+1\}$，$i=1,2,\ldots,N$；学习率$\eta$（$0<\eta \leq 1$）；

输出：a,b；感知机模型$f(x)=\mathrm{sign}\left(\sum_{j=1}^{N}\alpha_j y_j x_j \cdot x+b\right)$

其中$a=(a_1,a_2,\ldots,a_N)^T$。

（1）$a \leftarrow 0$，$b \leftarrow 0$

（2）在训练集中选取数据(x_i, y_i)

（3）如果 $y_i\left(\sum_{j=1}^{N}\alpha_j y_j x_j \cdot x_i+b\right) \leqslant 0$，$\alpha_i \leftarrow \alpha_i+\eta$，$b \leftarrow b+\eta y_i$

（4）转至（2）直到没有误分类数据。

Gram矩阵 $G=[x_i \cdot x_j]_{N\times N}$

例**2.2**　数据同例2.1，正样本点是$x_1=(3,3)^T$，$x_2=(4,3)^T$，负样本点是$x_3=(1,1)^T$，试用感知机学习算法对偶形式求感知机模型。

（1）取$a_i=0$，$i=1,2,3$，$b=0$，$\eta=1$

（2）计算Gram矩阵

$$\boldsymbol{G}=\begin{bmatrix}18 & 21 & 6\\ 21 & 25 & 7\\ 6 & 7 & 2\end{bmatrix}$$

（3）误分条件

$$y_i\left(\sum_{j=1}^{N}\alpha_j y_j x_j\cdot x_i+b\right)\leqslant 0$$

参数更新

$$\alpha_i\leftarrow\alpha_i+1,\quad b\leftarrow b+y_i$$

- i=1，样本点(x_1,y_1)
  a_1=0+1=1，b=0+1=1

$$y_1\left(\sum_{j=1}^{3} a_j y_j x_j \bullet x_1 + b\right) = a_1 y_1 x_1 \bullet x_1 + b = 19 > 0$$

- i=2，样本点(x_2,y_2)，a_1=1，b=1

$$y_2\left(\sum_{j=1}^{3} a_j y_j x_j \bullet x_2 + b\right) = a_1 y_1 x_1 \bullet x_2 + b = 22 > 0$$

- i=3，样本点(x_3,y_3)，a_1=1，b=1

$$y_3\left(\sum_{j=1}^{3} a_j y_j x_j \bullet x_3 + b\right) = -a_1 y_1 x_1 \bullet x_3 - b = -7 \le 0$$

a_3=a_3+1=1，b=1-1=0

（4）迭代。

（5）

$$w = 2x_1 + 0x_2 - 5x_3 = (1,1)^{\mathrm{T}}$$

$$b = -3$$

分离超平面 $x^{(1)} + x^{(2)} - 3 = 0$

感知机模型 $f(x) = \mathrm{sign}(x^{(1)} + x^{(2)} - 3)$

k	0	1	2	3	4	5	6	7
		x_1	x_3	x_3	x_3	x_1	x_3	x_3
α_1	0	1	1	1	2	2	2	2
α_2	0	0	0	0	0	0	0	0
α_3	0	0	1	2	2	3	4	5
b	0	1	0	-1	0	-1	-2	-3

原始形式和对偶形式的选择

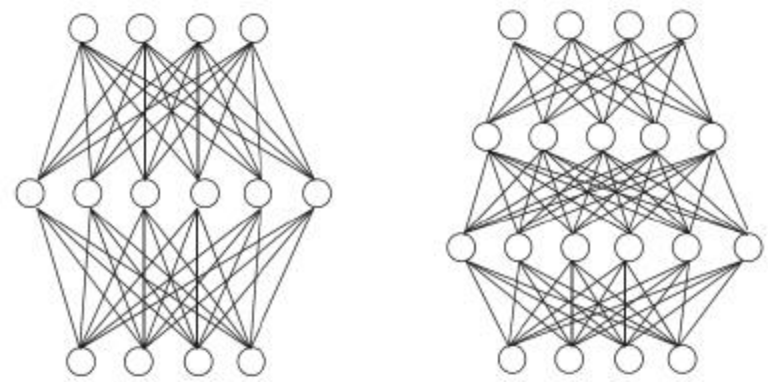
- 在向量维数（特征数）过高时，计算内积非常耗时，应选择对偶形式算法加速。
- 在向量个数（样本数）过多时，每次计算累计和就没有必要，应选择原始算法

2.4 误差逆传播算法 (BP)

多层前馈网络结构

多层网络：包含隐层的网络

前馈网络：神经元之间不存在同层连接也不存在跨层连接，即网络中无环或者回路。



(a) 单隐层前馈网络　　(b) 双隐层前馈网络

隐层和输出层神经元亦称“功能单元”(functional unit)，无隐藏层的又称“感知机(Perceptron)”

多层前馈网络有强大的表示能力

只需一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数

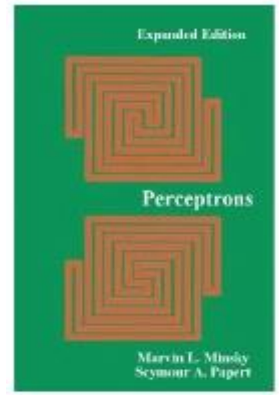
但是，如何设置隐层神经元数是未决问题. 实际常用“试错法”

神经网络发展回顾

1940年代 -萌芽期： M-P模型 (1943), Hebb 学习规则 (1945)

1958左右 -1969左右 ~繁荣期 ： 感知机 (1958), Adaline (1960), …

1969年： Minsky & Papert “Perceptrons”



冰 河期

1985左右 -1995左右 ~繁荣期 ： Hopfield (1983), BP (1986), …

1995年左右：SVM 及 统计学习 兴起

沉 寂期

2010左右 -至今 ~繁荣期 ：深度学习

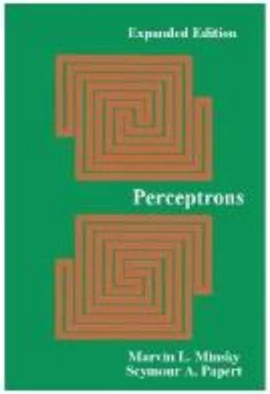
交替模式 ：
热十（年）
冷十五（年）

神经网络发展回顾

1940年代 -萌芽期： M-P模型 (1943), Hebb 学习规则 (1945)

1958左右 -1969左右 ~繁荣期 ： 感知机 (1958), Adaline (1960), …

1969年： Minsky & Papert “Perceptrons”



冰 河期

1985左右 -1995左右 ~繁荣期 ： Hopfield (1983), BP (1986), …

1995年左右：SVM 及 统计学习 兴起

沉 寂期

2010左右 -至今 ~繁荣期 ：深度学习

交替模式：
热十（年）
冷十五（年）

启示

科学的发展总是“螺旋式上升”

三十年河东、三十年河西

坚持才能有结果！

追热门、赶潮流 —— 三思而后行

误差逆传播算法 (BP)

最成功、最常用的神经网络算法，可被用于多种任务（不仅限于分类）

P. Werbos在博士学位论文中正式提出:

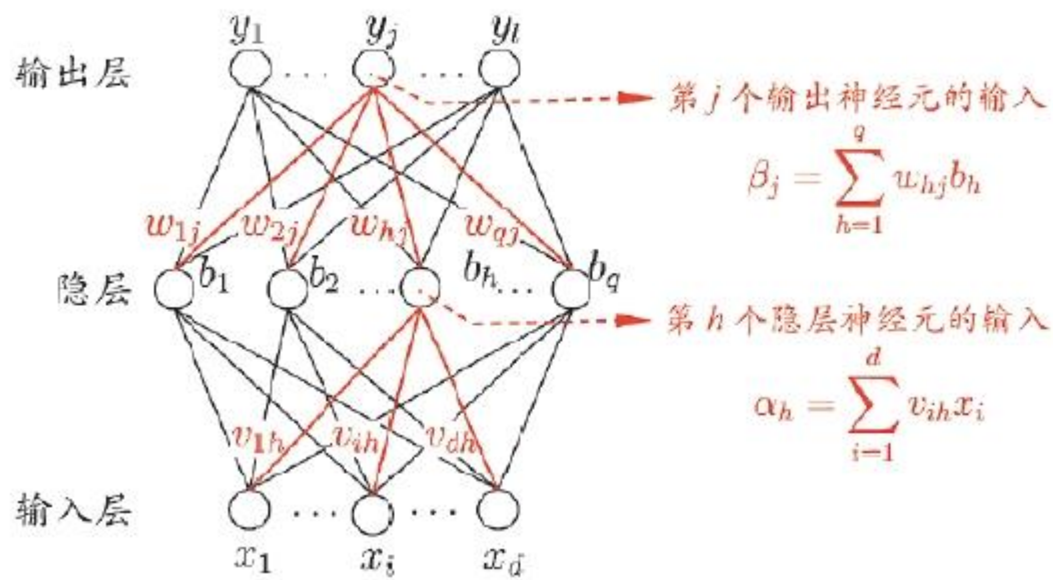
P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral science. Ph.D dissertation, Harvard University, 1974

给定训练集 $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_m, \boldsymbol{y}_m)\}, \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \mathbb{R}^l$

输入： d 维特征向量
输出： l 个输出值

隐层：假定使用 q 个隐层神经元

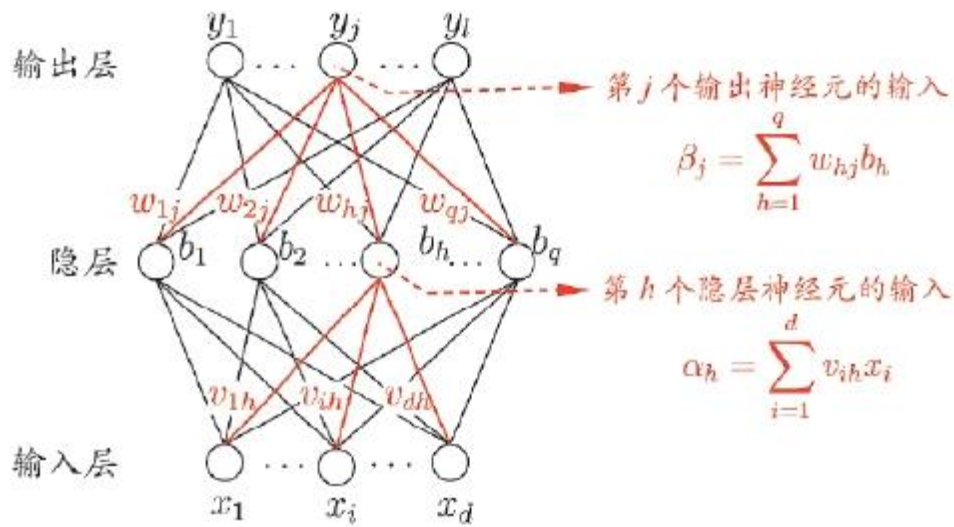
假定功能单元均使用 Sigmoid 函数



BP 算法推导

对于训练例 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, 假定网络的实际输出为 $\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$

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



则网络在 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ 上的均方误差为:

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$$

需通过学习确定的参数数目: $(d+l+1)q+l$

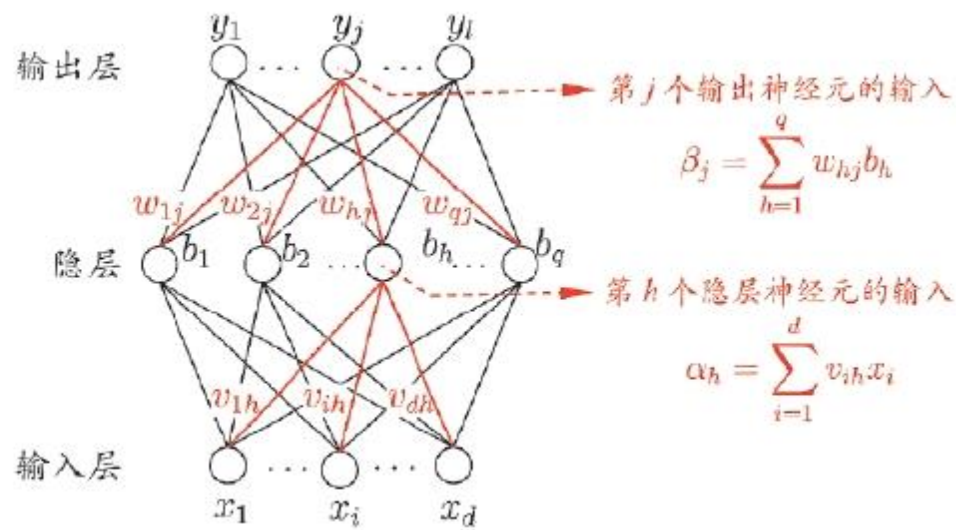
BP 是一个迭代学习算法，在迭代的每一轮中采用如下误差修正:

$$v \leftarrow v + \boxed{\Delta v}$$

BP 算法推导 (续)

BP 算法基于 梯度下降 策略，以目标的负梯度方向对参数进行调整

以 w_{hj} 为例



对误差 E_k, 给定学习率 η , 有:

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

注意到 w_{hj} 先影响到 β_j ,

再影响到 $\hat{y}_j^k$, 然后才影响到 E_k, 有 :

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

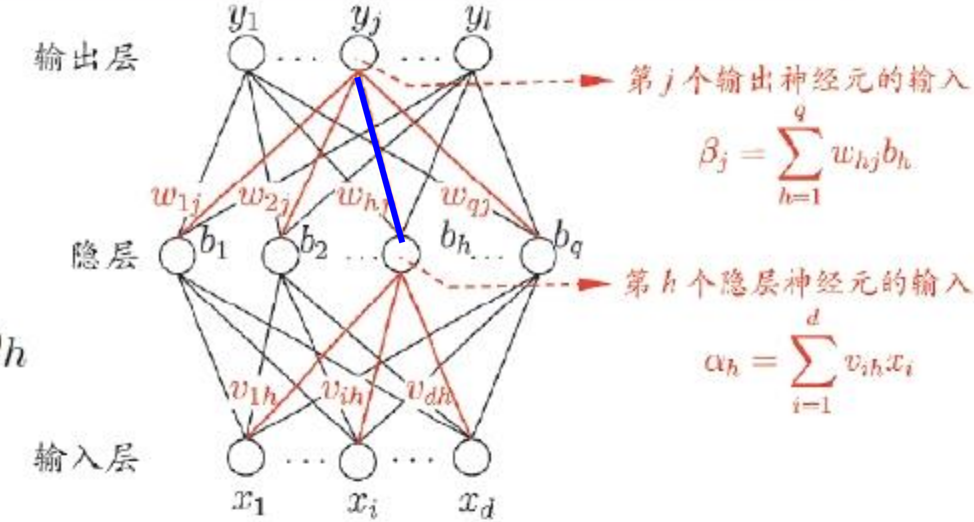
“链式法则”

BP 算法推导 (续)

$$y_j = f(\beta_j - \theta_j)$$

$$\frac{\partial E_k}{\partial w_{hj}} = \boxed{\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}} \cdot \boxed{\frac{\partial \beta_j}{\partial w_{hj}}}$$

$$= b_h$$



$$g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$$

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

$$= \underline{\hat{y}_j^k (1 - \hat{y}_j^k)} (y_j^k - \hat{y}_j^k) \ .$$

对 $\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$, 有

$$f'(x) = f(x)(1 - f(x))$$

再注意到 $\hat{y}_j^k = f(\beta_j - \theta_j)$

于是,

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} = \eta g_j b_h$$

BP 算法推导（续）

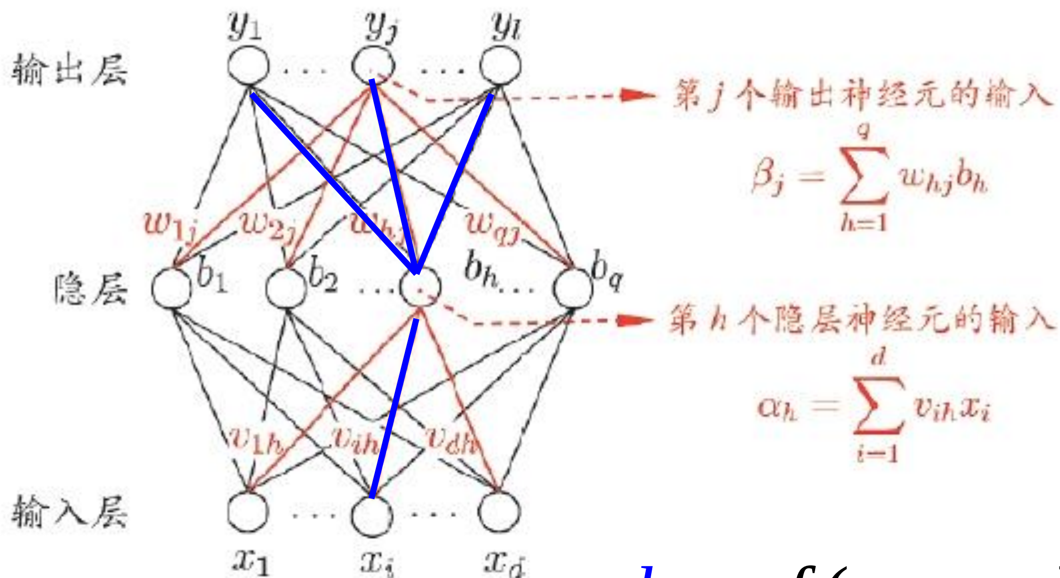
$$y_j = f(\beta_j - \theta_j)$$

类似地，有：

$$\Delta\theta_j = -\eta g_j$$

$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta\gamma_h = -\eta e_h$$



$$b_h = f(\alpha_h - \gamma_h)$$

其中：

$$e_h = -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h}$$

$$= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) = \sum_{j=1}^{l} w_{hj} g_j f'(\alpha_h - \gamma_h)$$

$$= b_h(1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$$

$\eta \in (0,1)$不能太大、不能太小

BP 算法

预处理：属性值一般伸缩到[-1,1], Y伸缩到[0,1]

输入： 训练集 $D = \{(\boldsymbol{x}_k, \boldsymbol{y}_k)\}_{k=1}^{m}$;
学习率 η.

过程:

```
在(0, 1)范围内随机初始化网络中所有连接权和阈值
repeat
    for all (x_k, y_k) ∈ D do
        根据当前参数和式(5.3) 计算当前样本的输出 ŷ_k;
        根据式(5.10) 计算输出层神经元的梯度项 g_j;
        根据式(5.15) 计算隐层神经元的梯度项 e_h;
        根据式(5.11)-(5.14) 更新连接权 w_hj, v_ih 与阈值 θ_j, γ_h
    end for
until 达到停止条件
```

输出： 连接权与阈值确定的多层前馈神经网络

图 5.8 误差逆传播算法

标准 BP 算法 vs. 累积 BP 算法

$$E_k = \sum_{j=1}^{l}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k = \frac{1}{2m}\sum_{k=1}^{m}\sum_{j=1}^{l}\left(\hat{y}_j^{(k)} - y_j^{(k)}\right)^2$$

标准 BP 算法

- 每次针对单个训练样例更新权值与阈值
- 参数更新频繁，不同样例可能抵消，需要多次迭代

累积 BP 算法

- 其优化目标是最小化整个训练集上的累计误差
- 读取整个训练集一遍才对参数进行更新，参数更新频率较低

在很多任务中，累计误差下降到一定程度后，进一步下降会非常缓慢, 这时标准 BP算法往往会获得较好的解, 尤其当训练集非常大时效果更明显.

缓解过拟合

BP算法常常导致过拟合

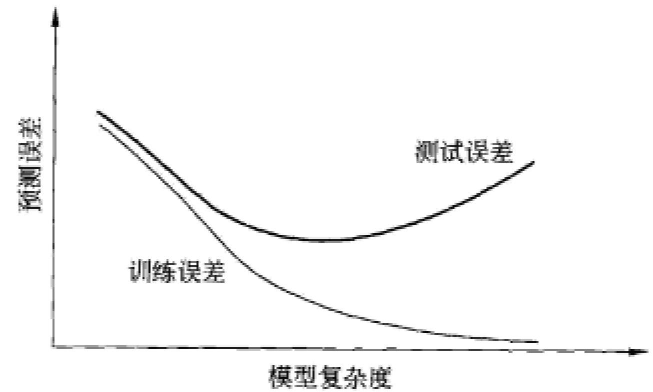


图 1.3　训练误差和测试误差与模型复杂度的关系

主要策略：

- **早停** (early stopping)
  - 若训练误差连续 a 轮的变化小于 b, 则停止训练
  - 使用验证集：若训练误差降低、验证误差升高，则停止训练

- **正则化** (regularization)
  - 在误差目标函数中增加一项描述网络复杂度

例如 $$E = \lambda \frac{1}{m} \sum_{k=1}^{m} E_k + (1-\lambda) \sum_i w_i^2$$

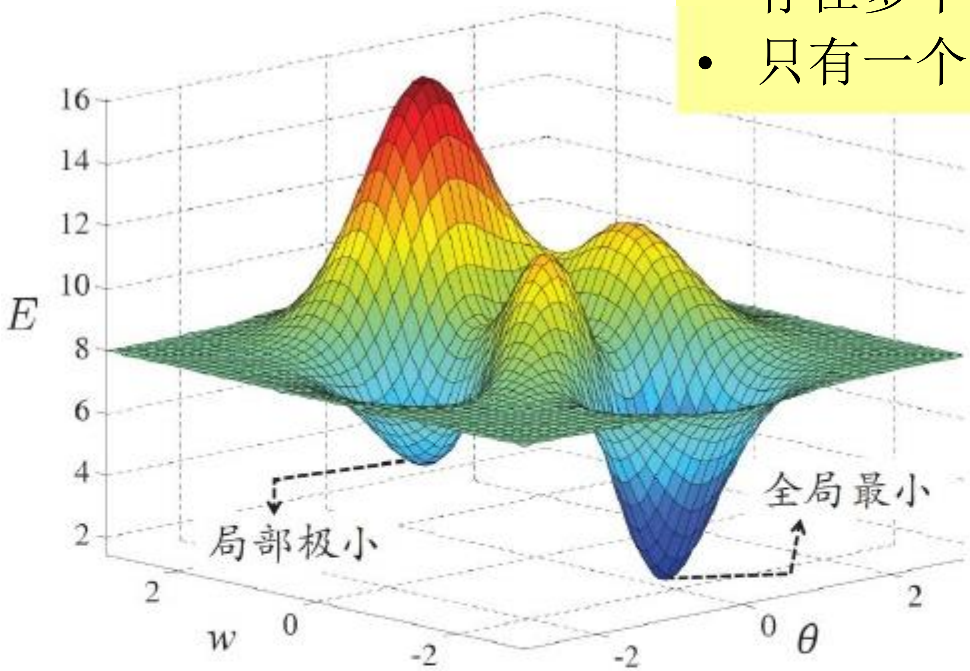
偏好比较小的连接权和阈值，
使网络输出更“光滑”

全局最小 vs. 局部极小

神经网络的训练过程可看作一个参数寻优过程：

在参数空间中，寻找一组最优参数使得误差最小

- 存在多个"局部极小"
- 只有一个"全局最小"



"跳出"局部极小的常见策略：

- ✓ 不同的初始参数
- ✓ 模拟退火
- ✓ 随机扰动
- ✓ 遗传算法
- ✓ ……

- 以多组不同参数值初始化多个神经网络, 按标准方法训练后, 取其中误差最小的解作为最终参数. 这相当于从多个不同的初始点开始搜索, 这样就可能陷入不同的局部极小, 从中进行选择有可能获得更接近全局最小的结果.

- 使用“模拟退火”(simulated annealing) 技术 [Aarts and Korst, 1989]. 模拟退火在每一步都以一定的概率接受比当前解更差的结果, 从而有助于“跳出”局部极小. 在每步迭代过程中, 接受“次优解”的概率要随着时间的推移而逐渐降低, 从而保证算法稳定.

- 使用随机梯度下降. 与标准梯度下降法精确计算梯度不同, 随机梯度下降法在计算梯度时加入了随机因素. 于是, 即便陷入局部极小点, 它计算出的梯度仍可能不为零, 这样就有机会跳出局部极小继续搜索.

此外, 遗传算法(genetic algorithms) [Goldberg, 1989] 也常用来训练神经网络以更好地逼近全局最小. 需注意的是, 上述用于跳出局部极小的技术大多是启发式, 理论上尚缺乏保障.

其他常见神经网络模型

- RBF： 分类任务中除BP之外最常用
- ART： “竞争学习”的代表
- SOM：最常用的聚类方法之一
- 级联相关网络：“构造性”神经网络的代表
- Elman网络：递归神经网络的代表
- Boltzmann机：“基于能量的模型”的代表
- ……