zhangy-lab


# 神经网络

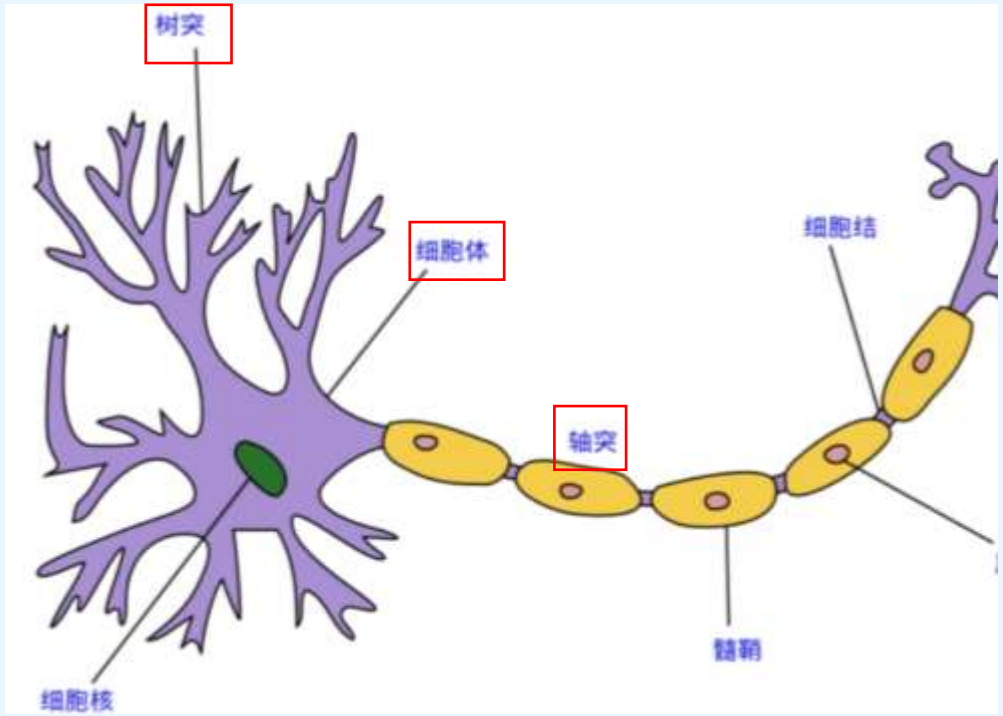
从“西瓜书”开始



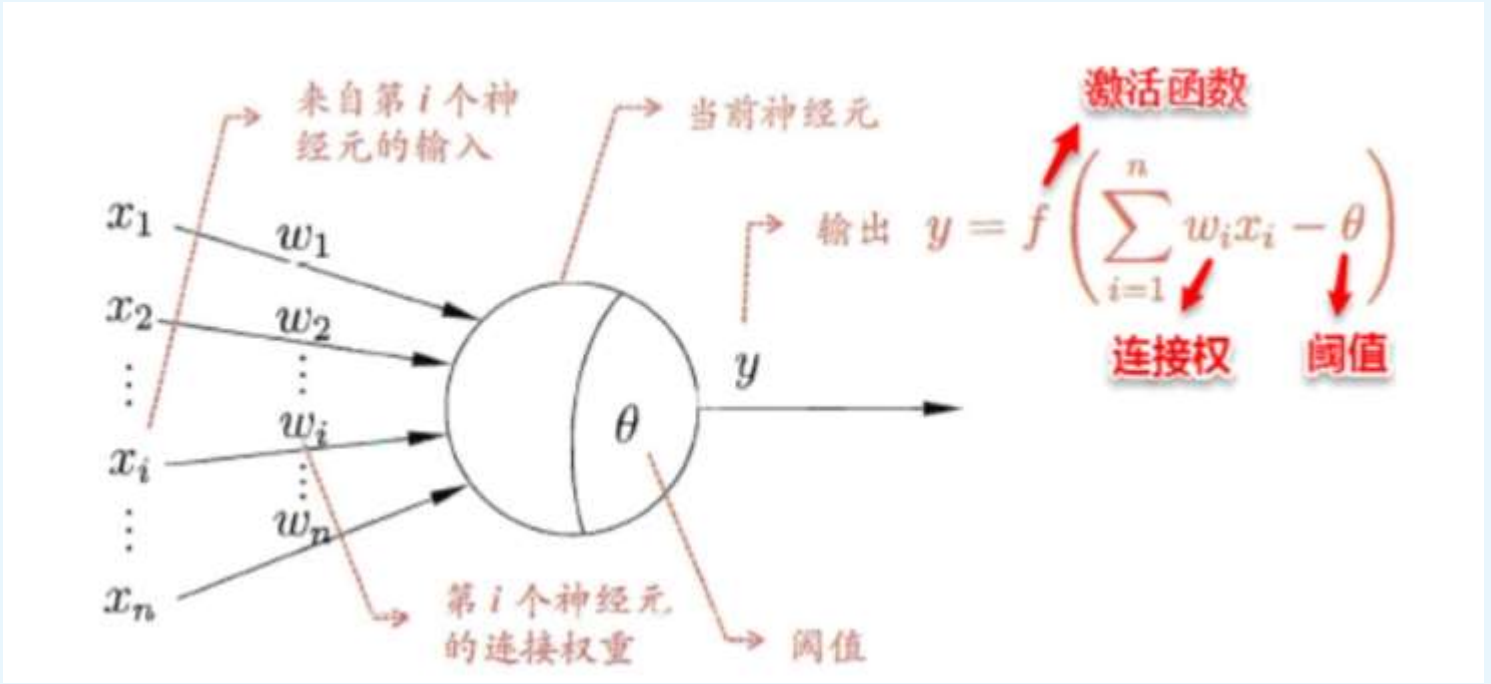
潘贤润
2024/06/30

# 5.1 神经元

神经网络：由具有适应性的简单单元组成的广泛并行互连的网络。



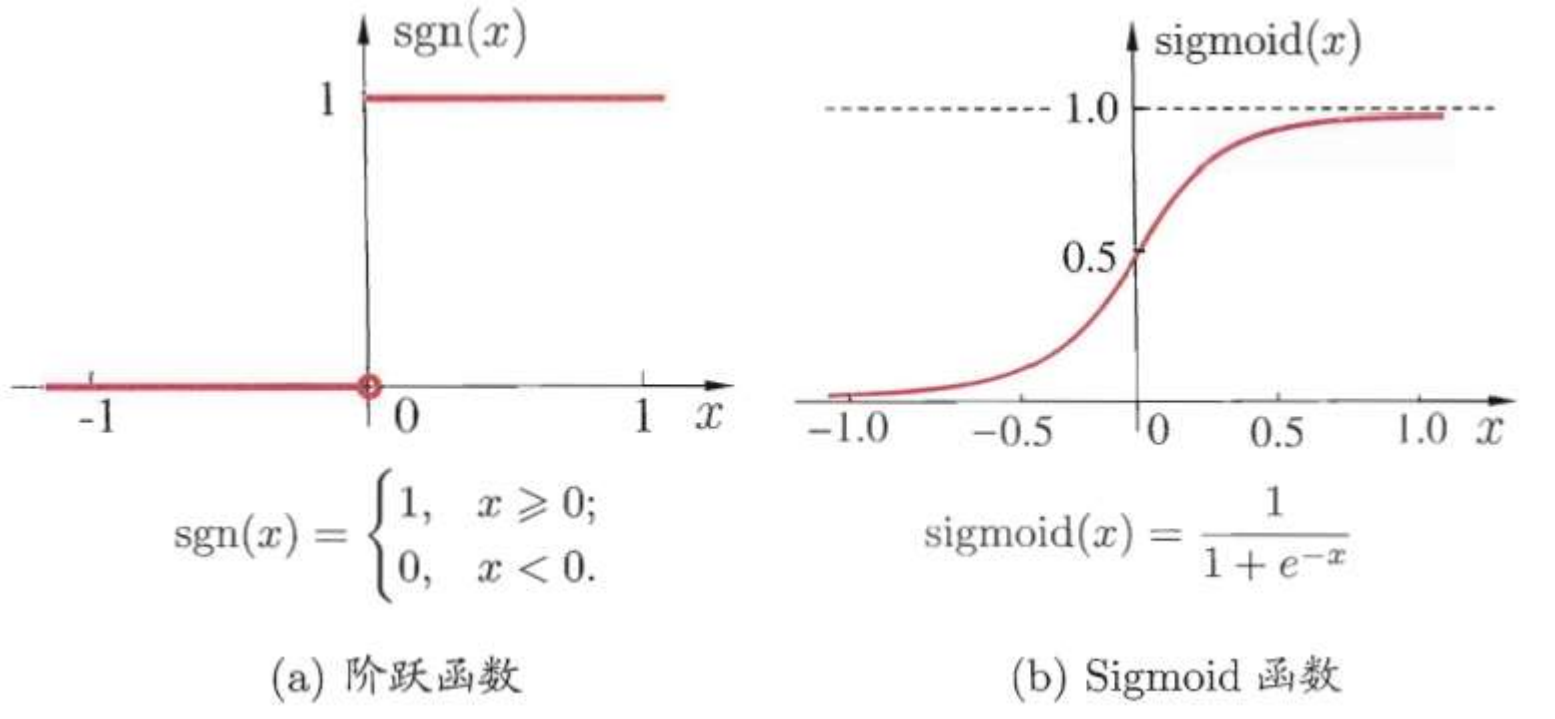

神经元模型（neuron）




M-P神经元模型

# 5.1 神经元

•激活函数/响应函数



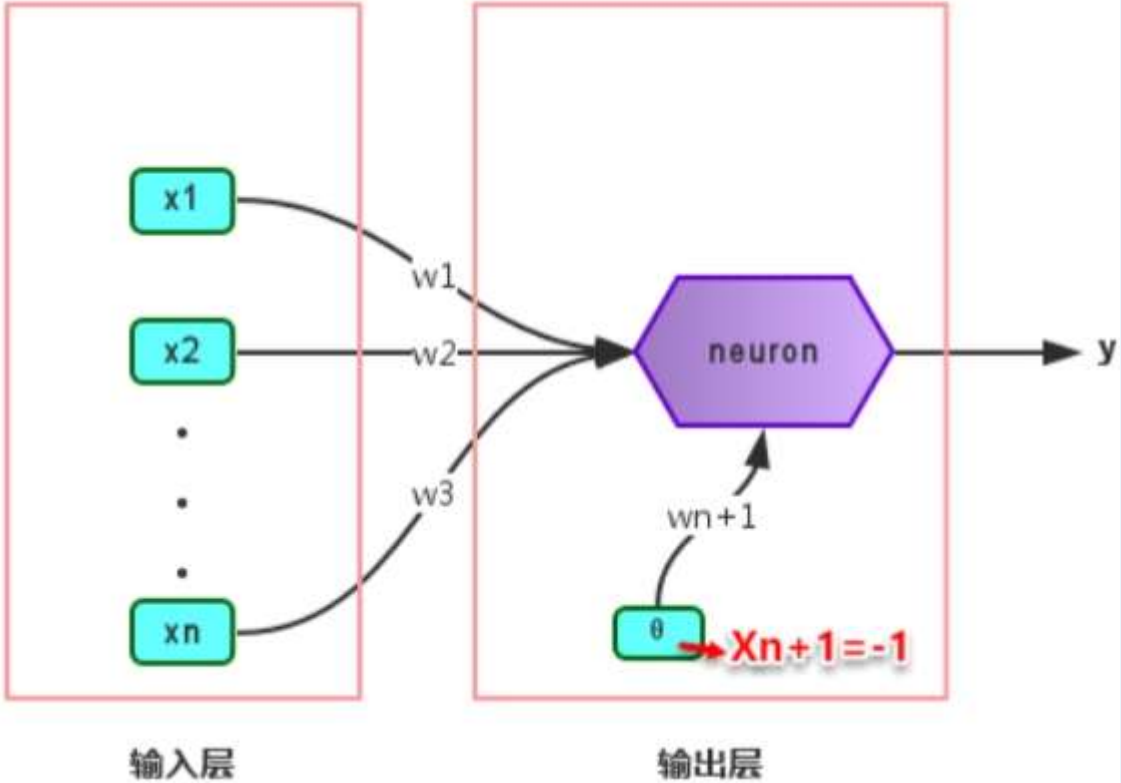

神经元模型最理想的激活函数也是阶跃函数，它将输入值映射为“0”（神经元抑制），“1”（神经元兴奋）。

Sigmoid函数将较大范围内变化的输入值挤压到 (0,1) 输出值范围内，所以也称为挤压函数（squashing function）。

将多个神经元按一定的层次结构连接起来，就得到了神经网络。若将每个神经元都看作一个函数，则整个神经网络就是由这些函数相互嵌套而成。

# 5.2 感知机与多层网络

•感知机




$$y = f(\sum_i w_i x_i - \theta)$$

感知机（Perceptron）是由两层神经元组成的一个简单模型，

给定训练集，则感知机的n+1个参数（n个权重+1个阈值）都可以通过学习得到。阈值Θ可以看作一个输入值固定为-1的哑结点的权重ωn+1，即假设有一个固定输入xn+1=-1的输入层神经元，其对应的权重为ωn+1，这样就把权重和阈值统一为权重的学习

$$y = f[\sum_n w_n x_n + (-1)\cdot w_{n+1}]$$
$$f(\sum_{n+1} w_{n+1} x_{n+1})$$

它不包含实际的数据，只是作为树结构中的一个占位符或者标记来使用。

# 5.2 感知机与多层网络

## •感知机训练

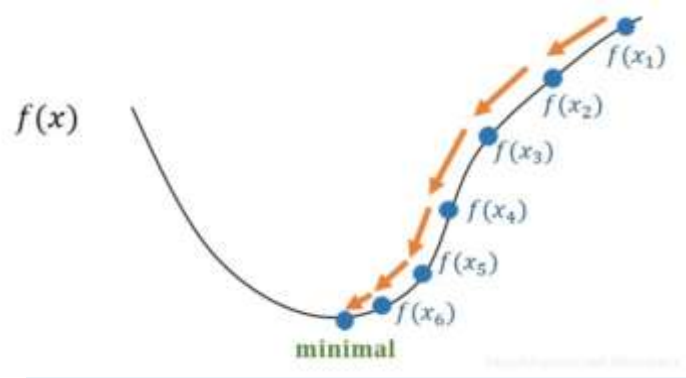
感知机权重的学习规则如下：对于训练样本（x，y），当该样本进入感知机学习后，会产生一个输出值，若该输出值与样本的真实标记不一致，则感知机会对权重进行调整，若激活函数为阶跃函数，则调整的方法为（基于梯度下降法）：

对于样本（x，y），其预测值为：

$$\hat{y}=f(\sum_{i=1}^{n}\omega_i x_i-\theta)=f(\sum_{i=1}^{n+1}\omega_i x_i),\text{其中}\ x_{n+1}=-1\text{为固定值}$$

均方误差为：$E=\frac{1}{2}(y-\hat{y})^2$

使用差来表示误差时，负误差和正误差会相互抵消




通过一点一点地调整自变量，以使目标函数值逐渐变小。这就好比你在山谷中寻找最低点的过程一样。

使用梯度下降法寻找最小的均方误差 $\min E$，负的梯度方向为最速下降方向

$$\frac{\partial E}{\partial \omega_i}=-(y-\hat{y})\frac{\partial \hat{y}}{\partial \omega_i}=-(y-\hat{y})\frac{\partial f(\sum_{i=1}^{n+1}\omega_i x_i)}{\partial \omega_i}$$

对权重 w 求损失函数的偏导数，从而得到它们对应的梯度。得到了关于权重 w 和偏置 b 的梯度，然后可以利用梯度进行参数更新，进行梯度下降的迭代过程。

因为函数 f 为阶跃函数，故有：$\frac{\partial f(\sum_{i=1}^{n+1}\omega_i x_i)}{\partial \omega_i}=x_i$

相当于ax对a求导只剩x

令下降步长为 $\eta$，$\eta\in(0,1)$，则：

其中 η∈（0，1）称为学习率

$$\Delta\omega_i=-\frac{\partial E}{\partial \omega_i}*\eta=\eta(y-\hat{y})x_i$$

下降步长就是确定每次迭代中参数更新的幅度。若步长太大则下降太快容易产生震荡，若步长太小则收敛速度太慢

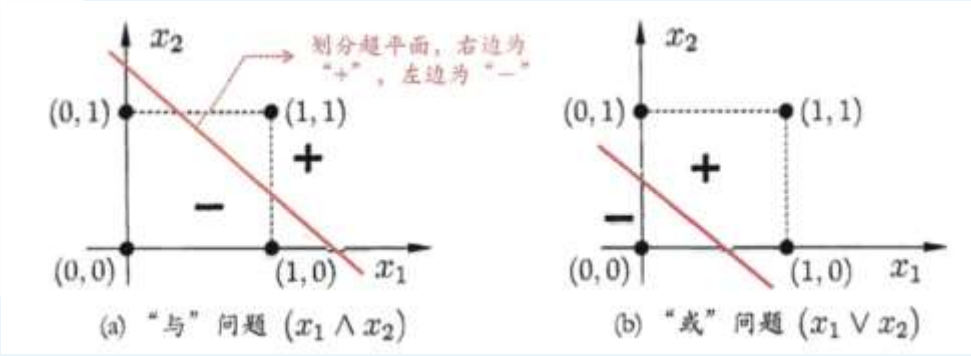
# 5.2 感知机与多层网络

•感知机解决的问题

“与” $(x_1 \wedge x_2)$: 令 $w_1 = w_2 = 1, \theta = 2$, 则 $y = f(1 \cdot x_1 + 1 \cdot x_2 - 2)$, 仅在 $x_1 = x_2 = 1$ 时, $y = 1$;

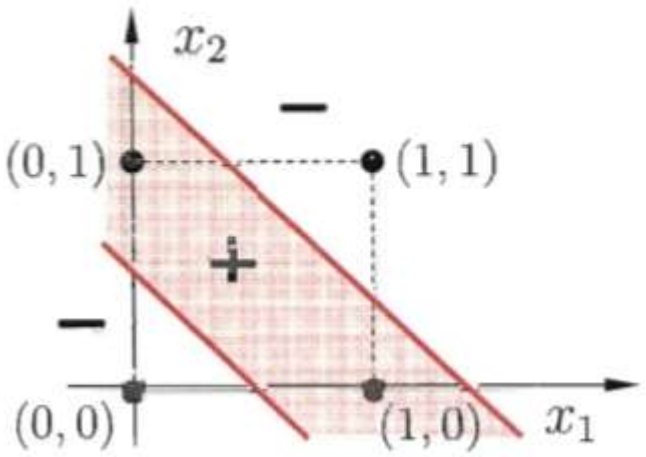
“或” $(x_1 \vee x_2)$: 令 $w_1 = w_2 = 1, \theta = 0.5$, 则 $y = f(1 \cdot x_1 + 1 \cdot x_2 - 0.5)$, 当 $x_1 = 1$ 或 $x_2 = 1$ 时, $y = 1$;

“非” $(\neg x_1)$: 令 $w_1 = -0.6, w_2 = 0, \theta = -0.5$, 则 $y = f(-0.6 \cdot x_1 + 0 \cdot x_2 + 0.5)$, 当 $x_1 = 1$ 时, $y = 0$; 当 $x_1 = 0$ 时, $y = 1$.



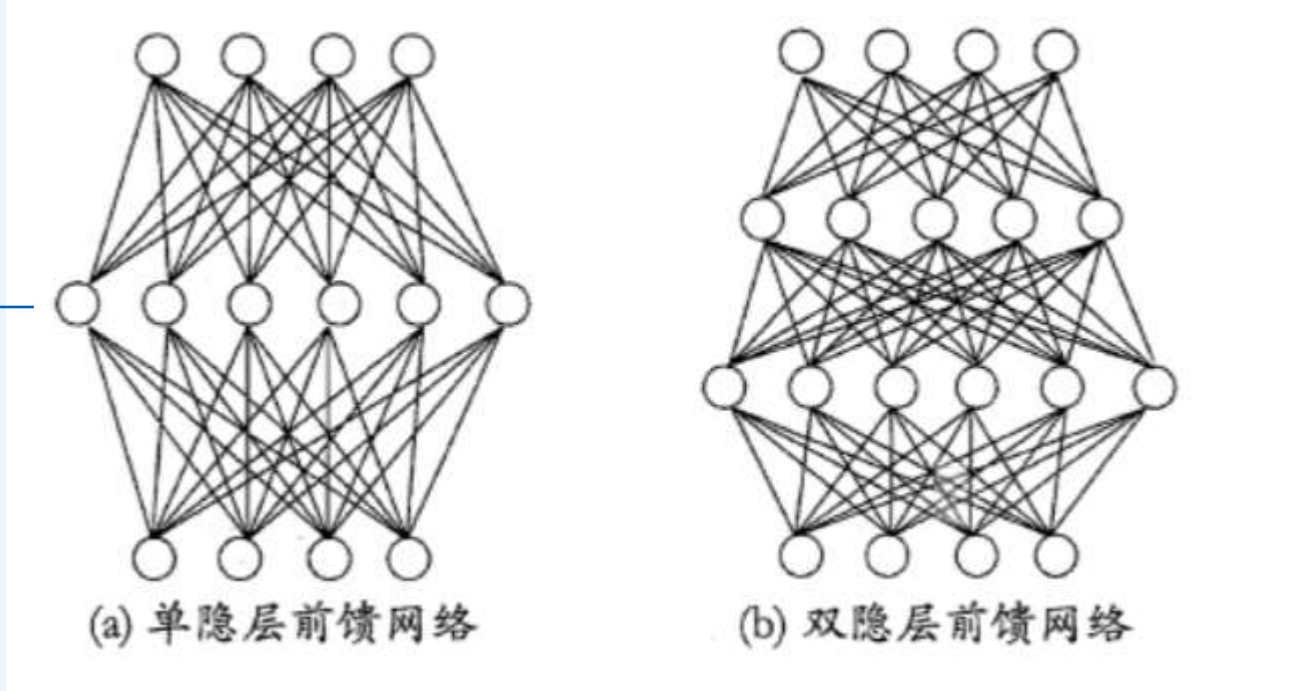

异或（XOR），它表示如果操作数之一是 true（1），而另一个是 false（0），则返回 true；否则返回 false。
常用符号为“⊕”。异或问题不能被单个线性决策边界所分割

| A | B | A ⊕ B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



# 5.2 感知机与多层网络

## •多层网络

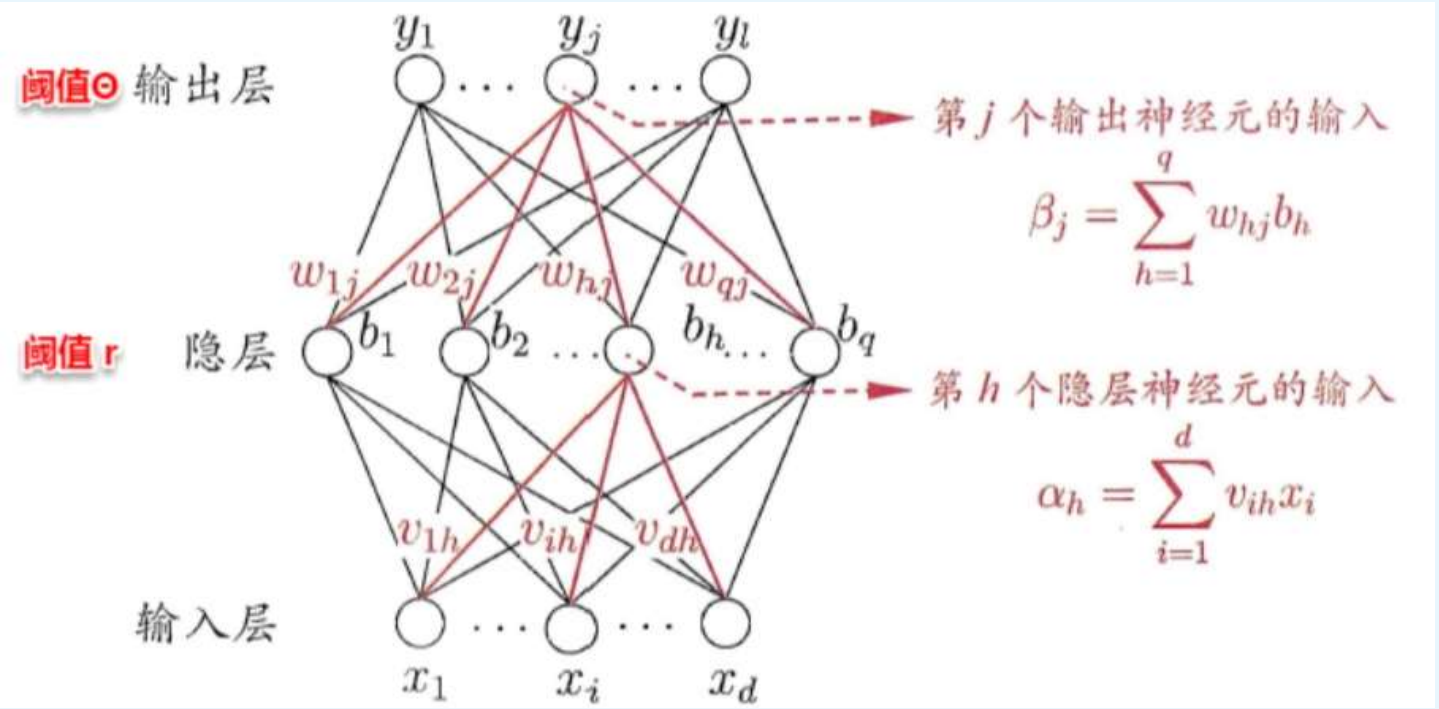



- 多层神经网络
- 多层前馈神经网络
  - * 每层神经元与下一层神经元之间完全互连
  - * 神经元之间不存在同层连接
  - * 神经元之间不存在跨层连接

这里的“前馈”指的是网络拓扑结构中不存在环或回路，而不是指该网络只能向前传播而不能向后传播

# 5.2误差逆传播算法

•BP神经网络算法




- 梯度下降法
- 均方误差
- 隐层→输出层
- 输入层→隐层

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

其中 $\hat{y}_j = f(\sum_{h=1}^{q} w_{hj} b_h - \theta_j)$ 第 $j$ 个神经元的输出

令 $b_{q+1} = -1$

$\hat{y}_j = f(\sum_{h=1}^{q+1} w_{hj} b_h)$ 令其 $= \beta_j$

$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$ Sigmoid 函数有一个很好的性质:

$= \frac{1}{2}(\hat{y}_j^2 - 2\hat{y}_j y_j + y_j^2) \cdot f'(\beta_j) \cdot b_h$ $f'(x) = f(x)(1-f(x))$

$= \frac{1}{2}(2\hat{y}_j - 2y_j + 0) \cdot [f(\beta_j)(1-f(\beta_j))] \cdot b_h$

$= (\hat{y}_j - y_j) \cdot \hat{y}_j \cdot (1-\hat{y}_j) \cdot b_h$

故 $\Delta w_{hj} = -\frac{\partial E_k}{\partial w_{hj}} \cdot \eta$, $\eta \in (0,1)$

$\Delta w_{hj} = \eta(y_j - \hat{y}_j) \cdot \hat{y}_j \cdot (1-\hat{y}_j) \cdot b_h$

$\Delta \theta_j = -\eta(y_j - \hat{y}_j) \cdot \hat{y}_j \cdot (1-\hat{y}_j)$ BP $b_h = -1$

同理:

$\Delta v_{ih} = \eta b_h (1-b_h) \sum_{j=1}^{l} w_{hj} g_j x_i$

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

# 5.2误差逆传播算法

•BP神经网络算法

BP算法的更新规则是基于每个样本的预测值与真实类标的均方误差来进行权值调节，即BP算法每次更新只针对于单个样例。需要注意的是：BP算法的最终目标是要最小化整个训练集D上的累积误差，即：

$$E = \frac{1}{m}\sum_{k=1}^{m} E_k \ ,$$

如果基于累积误差最小化的更新规则，则得到了累积误差逆传播算法，即每次读取全部的数据集一遍，进行一轮学习，从而基于当前的累积误差进行权值调整，因此参数更新的频率相比标准BP算法低了很多

**缓解BP网络的过拟合问题：**

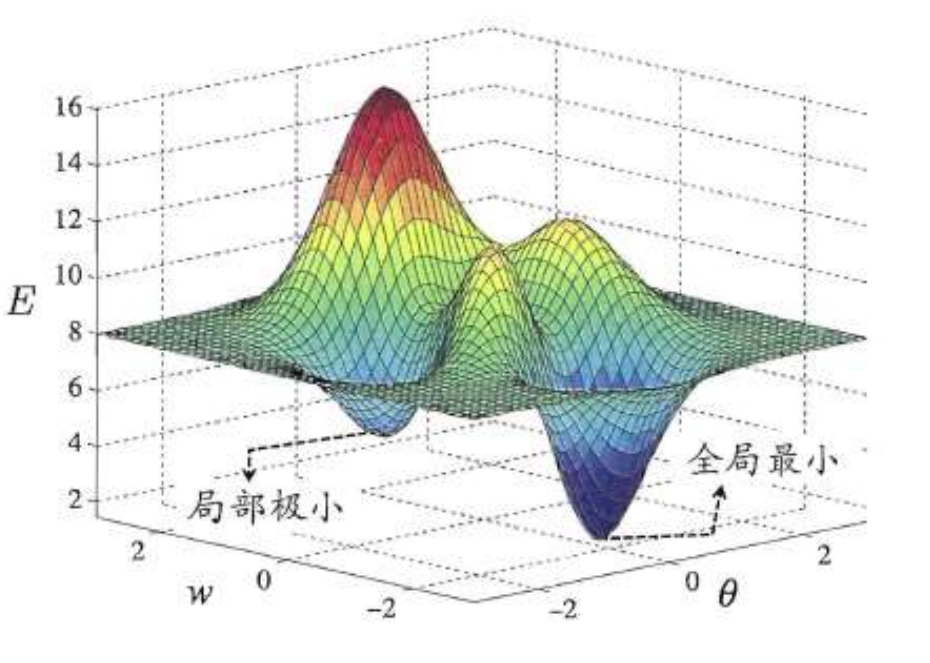
早停：将数据分为训练集与测试集，训练集用于学习，测试集用于评估性能，若在训练过程中，训练集的累积误差降低，而测试集的累积误差升高，则停止训练。

引入正则化：基本思想是在累积误差函数中增加一个用于描述网络复杂度的部分，例如所有权值与阈值的平方和，其中λ∈（0,1）用于对累积经验误差与网络复杂度这两项进行折中，常通过交叉验证法来估计。

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

# 5.2 全局最小与局部最小

•多层网络




- 局部极小解：参数空间中的某个点，其邻域点的误差函数值均不小于该点的误差函数值。
- 全局最小解：参数空间中的某个点，所有其他点的误差函数值均不小于该点的误差函数值。

◆ 以多组不同参数值初始化多个神经网络，按标准方法训练，迭代停止后，取其中误差最小的解作为最终参数。

◆ 使用“模拟退火”技术，通过在解空间中随机跳跃并逐渐减小“温度”（一个控制接受次优解概率的参数），从而逐步朝向最优解的方向搜索。

◆ 使用随机梯度下降，即在计算梯度时加入了随机因素，使得在局部最小时，计算的梯度仍可能不为0，从而迭代可以继续进行。

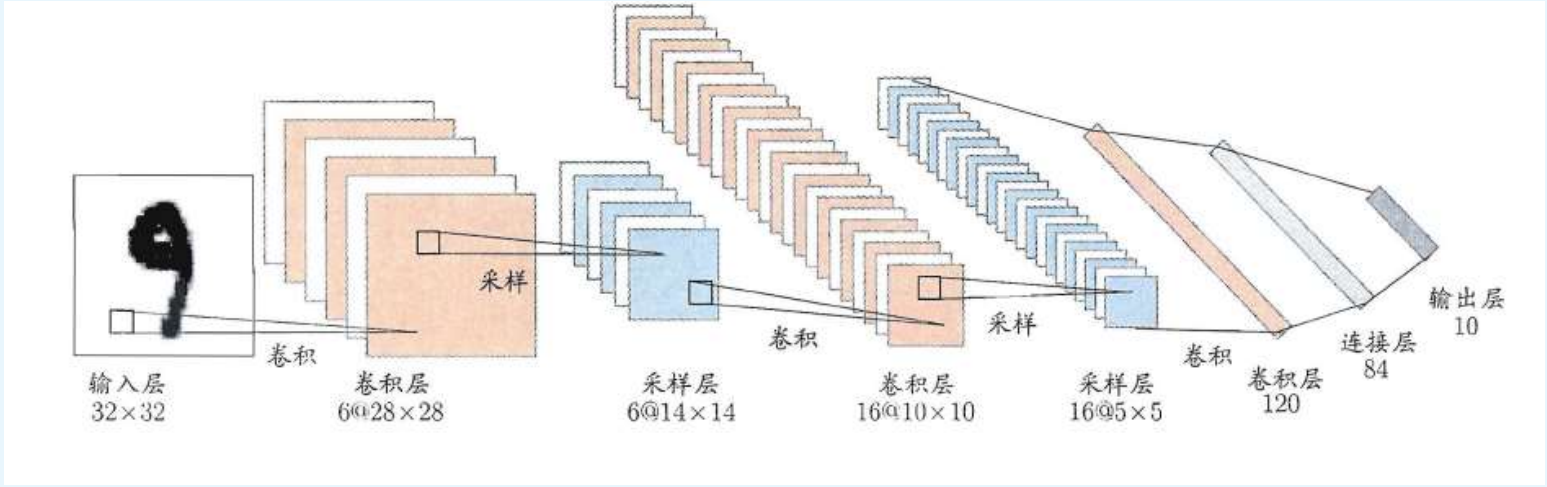
# 5.5 深度学习

理论上，参数越多，模型复杂度就越高，容量就越大，从而能完成更复杂的学习任务。

- 增加隐层的数目
- 增加隐层神经元的数目

无监督逐层训练：每次训练一层隐节点，把上一层隐节点的输出当作输入来训练，本层隐结点训练好后，输出再作为下一层的输入来训练，这称为预训练。全部预训练完成后，再对整个网络进行微调训练。把大量的参数进行分组，先找出每组较好的设置，再基于这些局部最优的结果来训练全局最优。

权共享：令同一层神经元使用完全相同的连接权，典型的例子是卷积神经网络。这样做可以大大减少需要训练的参数数目



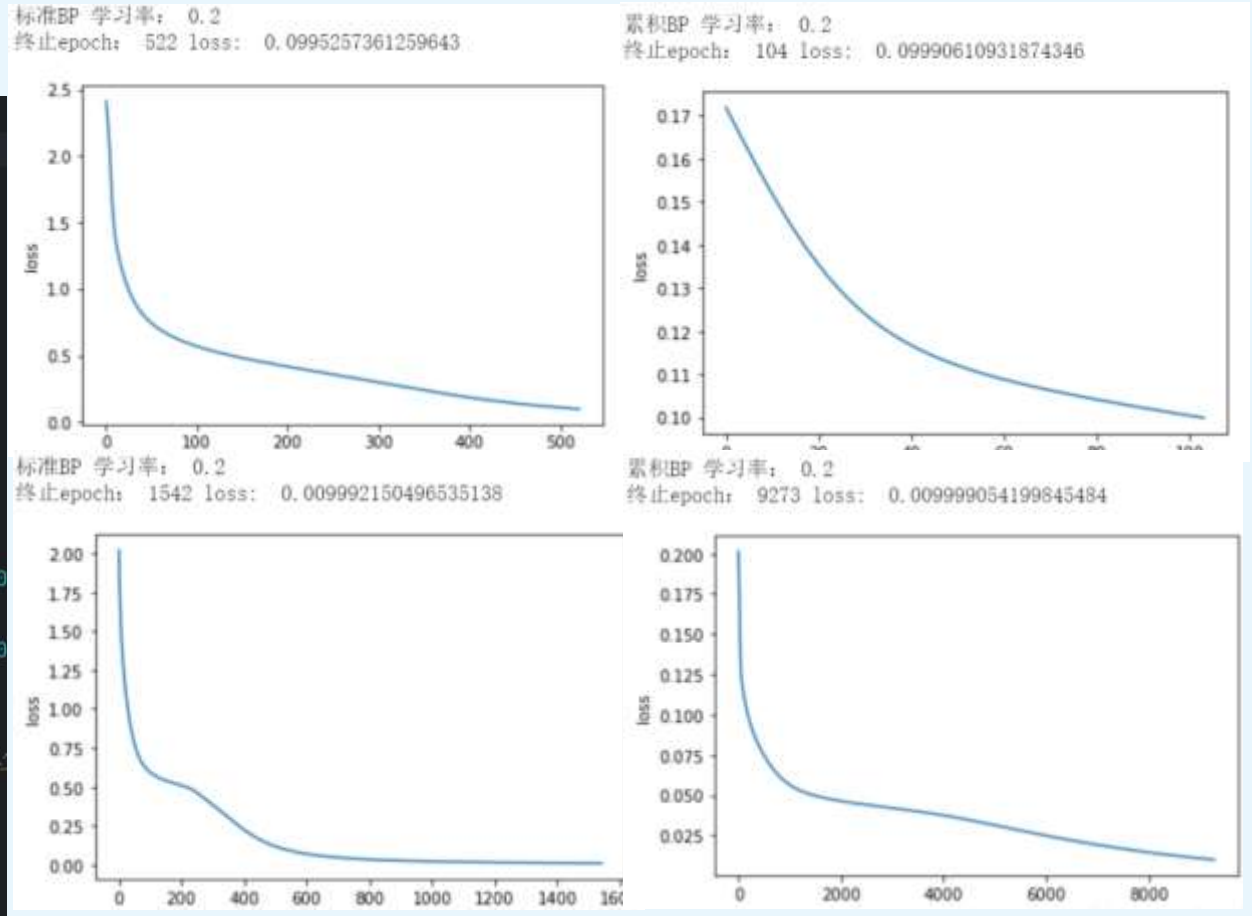

卷积神经网络的工作原理:
通过在图像上滑动卷积核，从而探测和提取图像中的各种特征。通过多次这样的观察和提取，它可以逐渐识别出图像中的更复杂的特征，最终做出正确的分类和识别。

# 5.6 课后习题

**5.5** 试编程实现标准 BP 算法和累积 BP 算法，在西瓜数据集 3.0 上分别用这两个算法训练一个单隐层网络，并进行比较.

```python
#NumPy库是用来进行科学计算和数据处理的Python库
import numpy as np
#Matplotlib库是一个用于数据可视化的Python库
import matplotlib.pyplot as plt

# 西瓜数据集 每一列为一条数据
features = np.array([
    [1, 2, 2, 1, 0, 1, 2, 2, 2, 1, 0, 0, 1, 0, 2, 0, 1],
    [2, 2, 2, 2, 2, 1, 1, 1, 1, 0, 0, 2, 1, 1, 1, 2, 2],
    [1, 0, 1, 0, 1, 1, 1, 1, 0, 2, 2, 1, 1, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 2, 2, 1, 1, 0, 2, 1],
    [2, 2, 2, 2, 2, 1, 1, 1, 1, 0, 0, 0, 2, 2, 1, 0, 1],
    [1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1],
    [0.697, 0.774, 0.634, 0.608, 0.556, 0.403, 0.481, 0.437, 0.666, 0.243, 0.245, 0.343, 0.639, 0
     0.719],
    [0.460, 0.376, 0.264, 0.318, 0.215, 0.237, 0.149, 0.211, 0.091, 0.267, 0.057, 0.099, 0.161, 0
     0.103]
])
#这段代码创建了一个名为labels的NumPy数组，数组的元素是一个包含16个值的一维数组。这些值代表了某个分类问题的标签，其中1
labels=np.array([
    [1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0]
])
#1/(1+exp(-X)) 是 S 型函数（Sigmoid 函数）的定义
2 usages
def sigmoid(X):
    return 1./(1+np.exp(-X))
```




累积BP算法训练过程中的权重更新更加平滑

当累计误差下降到一定程度时，累积BP比标准BP慢，比如停止条件为loss<=0.01时

# 感谢倾听

See you next.