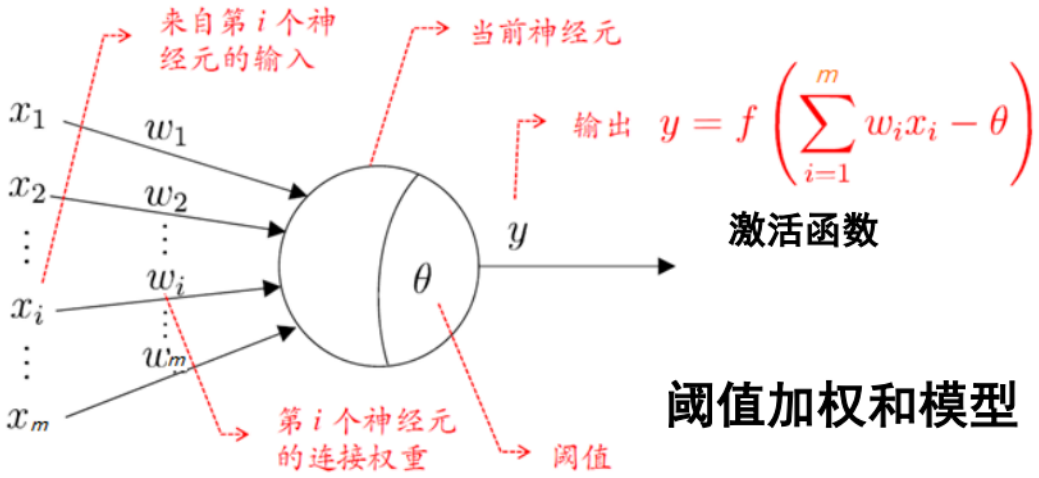
第五章：神经网络

神经网络：

定义：由具有适应性的简单单元组成的广泛并行互联的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的反应。

```
#### M-P神经元模型
```



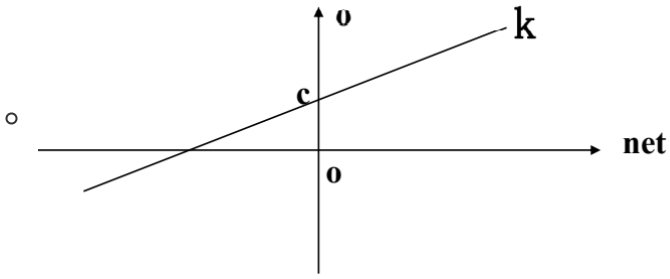
输入：来自其他m个神经元传递过来的输入信号

处理：输入信号通过带权重的连接进行传递，神经元接受到总输入值将与神经元的阈值进行比较

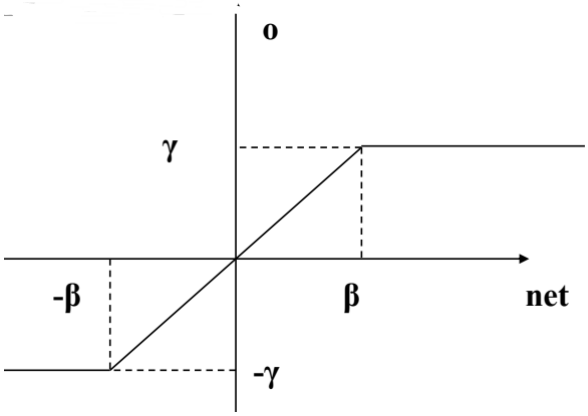
输出：通过激活函数的处理得到输出

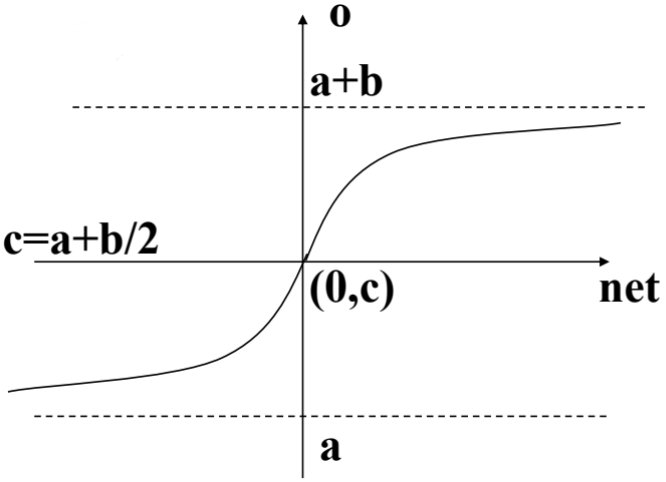
激活函数：Activation Function 对神经元所获得的网络输入进行变换。常用有：线性函数、非线性斜面函数、阈值函数、逻辑斯蒂函数

- 线性函数 Linear Function
  - $f(net) = k * net + c$
  - 只能进行线性变化，不适合处理非线性问题

- 

- 非线性斜面函数 Ramp Function
  - $f(net)=\begin{cases}\gamma & net\geq\beta\\ k*net & |net|<\beta\\ -\gamma & net\leq\beta\end{cases}$
  - γ 为大于0 的常数，被称为饱和值是神经元的最大输出

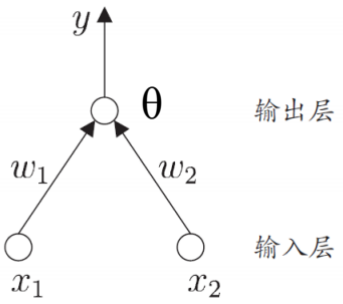


- 阈值函数 （Threshold Function）
  - $f(net)=\begin{cases}\alpha & net>\beta\\ -\gamma & net\leq\beta\end{cases}$
  - α、β、γ 均为 非负数，θ为阈值
- 逻辑斯蒂函数 Logistic Function
  - $f(net)=\frac{a+b}{1+e^{(-d*net)}}$
  - a，b，d为常数，该函数饱和值为a+b（极大），和 a(极小) （规定如是）
  - 当a=0,b=1,d=1时为最简形式：$f(net)=\frac{1}{1+e^{-net}}$
  - 

  - 逻辑斯蒂函数形如S也称S形函数；将较大范围变化的输入值挤压到（0，1）的输出值范围内，因此也称挤压函数；有较好的增益控制能力，也叫增益控制函数
  - 当使用激励函数是逻辑斯蒂函数时，此时的M-P神经元模型就是一个逻辑斯蒂回归模型

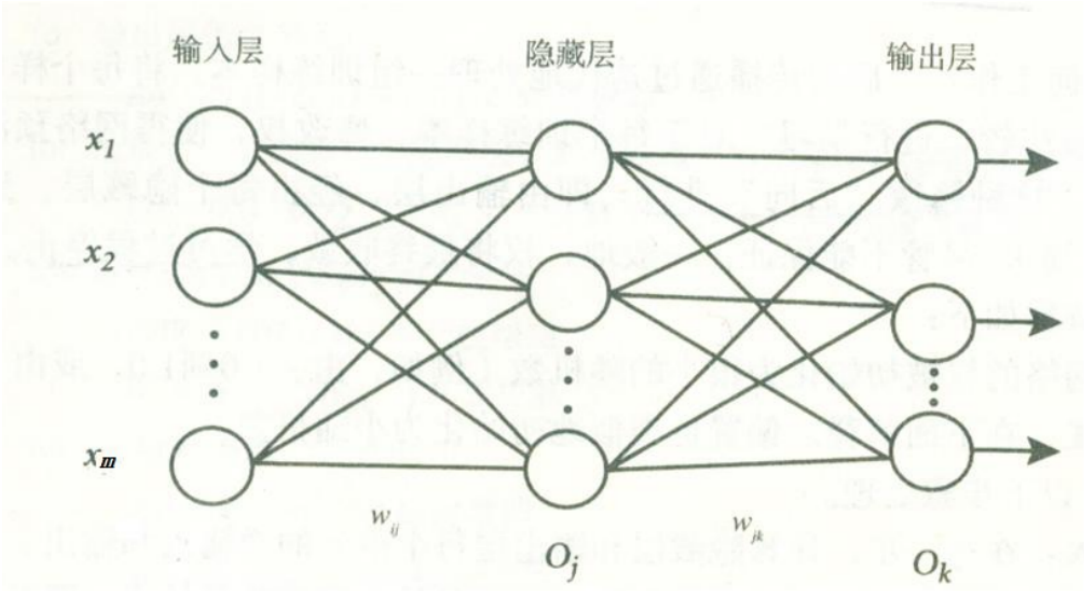
单层感知机

- 定义：只拥有一层M-P神经元的神经网络模型
- 如下图，最简单的单层感知机，输出层只有一个神经元



- 学习规则
  - 神经元的阈值θ看作一个固定输入为-1的'哑节点',所对应的连接权重ω_{m+1},这样权重和阈值的学习可统一为权值的学习了。
  - 对训练样例(x,y),若当前感知机输出为 $\hat{y}$,则感知机权重调整规则为：$\omega_i \leftarrow \omega_i + \Delta\omega_i$
    $$\Delta\omega_i = (l)(y-\hat{y})x_i$$
  - 其中 $l \in (0,1)$ 为学习率
  - 若感知机预测正确 $y=\hat{y}$则感知机权重不发生变化，否则根据二者差值进行权重调整
- 学习能力：
  - 单层感知机能非常容易地实现逻辑与、或、非运算。但学习能力有限
  - 若二分类问题是线性可分的，则感知机的学习过程一定会收敛，否则感知机学习会发生震荡。例如无法解决异或
  - 要解决非线性可分问题，需要使用多层M-P神经元

多层前馈神经网络

- **多层**指除了输入层和输出层之外还存在一个或多个隐含层
- **前馈**指：外界信号从输入层，经由隐含层到达输出层，不存在信号的逆向传播
- **全互连接**指：每层神经元与下层神经元全互连接，神经元之间不存在同层连接，也不存在跨层连接。
- 

多层前馈网的结构图

- 输入层不进行处理，只接受输入。隐含层与输出层进行函数处理
- 神经网络参数学习就是根据训练数据调整神经元之间的连接权以及每个功能神经元的阈值。

BP算法 Error Back Propagation，误差逆传播

- BP算法可用于多层前馈神经网络和其他类型神经网络例如递归神经网络，通常BP算法特指多层前馈神经网络的训练算法
- 训练过程
  - 1.初始化：神经网络的权和偏置
    - 输入层到隐含层的连接权矩阵W_{ij}
    - 隐含层到输出层 的连接权矩阵W_{jk}
    - 隐含层每个M-P神经元的阈值Θ_j
    - 输出层每个M-P神经元的阈值Θ_k
  - 2.进行反复调整，分为三个阶段
    - 信号的前向传播：在这个阶段要求计算出隐含层和输出层种每一个神经元的网络净输入和网络输出
    - 误差的逆向传播阶段：在这个阶段要求计算出输出层和隐含层中每一神经元的误差
    - 权值和阈值的更新阶段：该阶段要求更新所有连接权的权值和所有M-P神经元的阈值

1）初始化神经网络的权和偏置。

2）对实例中的每个训练样本：

① 计算每个隐含层节点的输入：$I_j = \sum_i X_i W_{ij} - \theta_j$

② 计算每个隐含层节点的输出：$O_j = 1/(1+e^{-I_j})$

③ 计算每个输出层节点的输入：$I_k = \sum_j O_j W_{jk} - \theta_k$

④ 计算每个输出层节点的输出：$O_k = 1/(1+e^{-I_k})$

⑤ 计算每个输出层节点的误差：$Err_k = O_k(1-O_k)(T_k - O_k)$

⑥ 计算每个隐含层节点的误差：$Err_j = O_j(1-O_j)\sum_k W_{jk} Err_k$

⑦ 更新输入层到隐含层的权值：$W_{ij} = W_{ij} + (l) X_i Err_j$

⑧ 更新隐含层到输出层的权值：$W_{jk} = W_{jk} + (l) O_j Err_k$

⑨ 更新每个隐含层节点的偏置：$\theta_j = \theta_j - (l) Err_j$

⑩ 更新每个输出层节点的偏置：$\theta_k = \theta_k - (l) Err_k$

3）判断终止条件是否满足，若满足则停止，否则重复执行步骤 2）。

- BP算法梯度下降策略推导为：
-

No
Date.

对训练例 (x_k, y_k)，假定神经网络输出为 $\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k \dots \hat{y}_l^k)$

即：$\hat{y}_j^k = f(\beta_j - \theta_j)$ ……①

则：在 (x_k, y_k) 上均方误差为：$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)^2$ ……②

给定学习率对误差 E_k 有：$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$，$\eta$ 为学习率 …③

w_{hj} 先影响到第 j 个神经元的输入值 β_j，再影响到输出值 $\hat{y}_j^k$，于是有：

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} \quad \text{……④}$$

依：$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$：

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h \quad \text{…⑤}$$

由于 Sigmod 函数：$\frac{1}{1+e^{-x}}$ 有个极好性质：

$$f'(x) = f(x)(1 - f(x)) \quad \text{……⑥}$$

于是：

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

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

$$= -(y_j^k - \hat{y}_j^k)\hat{y}_j^k(1 - \hat{y}_j^k) \quad \text{…⑦}$$

即代入①、⑤入④有：$\Delta w_{hj} = \eta g_j b_h$.

- BP算法存在的问题：
  - 结构学习问题：多层前馈神经网络输入层的神经元个数由输入数据的维度（连续属性）和编码方法（离散属性）确定；输出层神经元的个数由待分类的类别数目和编码方法确定。比如四分类问题需要编码两个神经元。
    - 研究表明，只要一个包含足够多神经元的隐含层，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。层数只要两层就可以
    - 如何确定隐含层神经元的个数是个未解的问题，实际应用中通常根据经验或者试错法调整。
  - 初始化问题：连接权和偏置都是初始化为不同的小随机数。“不同”保证网络可以学习；小随机数可以防止其值过大而提前进入饱和状态达到局部最小值

        - 其解决办法可以使用：重新初始化
    - 步长设置问题：学习率控制着算法每一轮迭代的更新步长。步长太小收敛速度过慢，步长太大又可能导致网络的不稳定、甚至瘫痪。
- 合适的方法是设置自适应步长来达到：训练开始时步长较大，随着训练进行步长逐步缩小的目的。
- 权值和阈值的更新问题：
    - 基本BP算法采用样例更新，即每处理一个训练样例就更新一次权值和阈值。
        - 这样的缺陷是：
            - 参数更新频繁、不同样例可能抵消、需要迭代次数较多
        - 训练样例的输入顺序对训练结果有较大影响，它更“偏爱”后输入样例
        - 解决方法：周期更新
        - 每处理一遍所有的训练样例才更新一次权值与阈值。
- 过拟合问题
    - 神经网络表示能力国强，经常遭遇过拟合。具体表现为：训练误差持续降低，测试误差继续上升。
    - 缓解过拟合问题的策略是：早停和正则化。早停是在训练过程中，若训练误差降低，但验证误差升高，则停止训练。正则化是在误差目标函数中增加一项描述网络复杂程度的部分，比如连接权值或阈值的平方和。$$E=\lambda\frac{1}{m}\sum_{k=1}^{m}E_k+(1-\lambda)\sum_i w_i^2\ ,$$
        - 其中$\lambda\in(0,1)$用于对经验误差和网络复杂度两项进行折中，通常交叉验证法来估计。

全局最小和局部最小

全局最小一定是局部最小，反之不然。基于梯度的搜索的方法中，我们从某点出发迭代寻找最优参数值。每次迭代时先计算误差函数在当前点的梯度，然后根据梯度确定搜索方向。若误差函数在当前点梯度为零，则达到局部最小。这时就会停止迭代，参数更新停止。因此可能只找到局部最小，而非全局最小。如图：

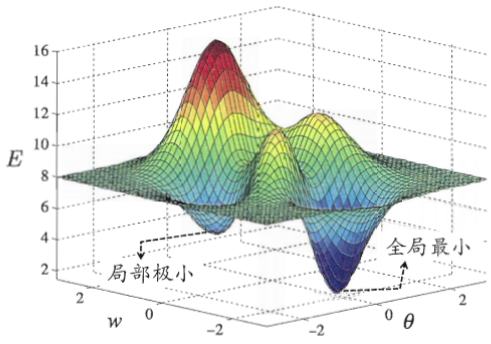


图 5.10 全局最小与局部极小

解决方法有：

- **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。这相当于从多个点开始搜索
- **模拟退火（simulate annealing）**：每一步都以一定概率接受比当前解更差的结果，从而有助于跳出"局部最小"。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来保证算法稳定。
- **随机梯度下降**：在计算梯度时加入随机因素。即便陷入局部极小点，计算出的梯度仍不为零，有机会跳出局部最小继续搜索。该方法理论上尚缺乏保障。

其他神经网络

- RBF网络：RBF（radial basis function，径向基函数）网络是一种单隐层前馈神经网络，它使用径向基作为隐层神 经元激活函数。其输出是对隐层神经元输出的线性组合。
- SOM网络：SOM（self-organizating map，自组织映射网络）是一种竞争学习型的无监督神经网络，它能够将高维 输入数据映射到低纬度空间，同时保留输入数据在高维空间的拓扑结构，即将高维空间的相似样本映射 到网络输出层。
- Elman网络：与前馈神经网络不同，递归神经网络（recurrent neural networks）允许网络中出现环形结构，从而使 得一些神经元的输出反馈作为输入信号。这样的信息反馈过程使得 t 时刻的输出不但与 t 时刻的输入有 关，还与 t-1 时刻的网络状态相关，从而能处理与时间有关的动态变化。 Elman网络是其中代表之一。
- 级联相关网络：一般的神经网络结构是固定的，训练目的是寻找合适的连接权重和阈值等参数。但结构自适应网络则是 将网络的结构也作为学习目标之一。级联相关网络就是其中的代表。
- ART网络：是竞争型学习的重要代表。该网络由比较层、识别层、识别阈值和重置模块构成。
    - 竞争最简单的方式是：计算输入向量和每个识别层神经元所对应的模式类的代表向量之间的距离，距离最小者获胜。获胜神经元向其他识别层神经元发送信号，抑制其激活。

- Boltzmann机：
-

> 神经网络中有一类模型是为网络状态定义一个“能量”(energy), 能量最小化时网络达到理想状态, 而网络的训练就是在最小化这个能量函数. Boltzmann 机 [Ackley et al., 1985] 就是一种“基于能量的模型”(energy-based model), 常见结构如图 5.14(a) 所示, 其神经元分为两层: 显层与隐层. 显层用于表示数据的输入与输出, 隐层则被理解为数据的内在表达. Boltzmann 机中的神经元都是布尔型的, 即只能取 0、1 两种状态, 状态 1 表示激活, 状态 0 表示抑制. 令向量 $\boldsymbol{s} \in \{0,1\}^n$ 表示 n 个神经元的状态, w_{ij} 表示神经元 i 与 j 之间的连接权, θ_i 表示神经元 i 的阈值, 则状态向量 $\boldsymbol{s}$ 所对应的 Boltzmann 机能量定义为
>
> $$E(\boldsymbol{s}) = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}s_i s_j - \sum_{i=1}^{n}\theta_i s_i \ . \tag{5.20}$$

深度学习

- 深层神经网络：多隐层神经网络
    - 增加模型宽度：增加隐含层神经元数目
    - 增加模型深度：增加隐含层的数目
    - 增加深度比增加宽度更有效，可以嵌套更多层的激活函数。但是多隐层神经网络难以直接使用经典算法（例如BP算法）进行训练，因为误差在多隐层内逆传播时往往会发散（diverge）而不饿能收敛
- 学习有效的深层神经网络方法：
    - 预训练+微调
        - 无监督逐层训练是多隐层网络训练的有效手段，基本思想是每次训练一层隐结点。训练时将上层隐结点的输出作为输入，而本层隐结点输出作为下一层的输入。这叫做预训练。
        - 训练完成后对整个网络进行微调：fine-turning，微调一般使用BP算法。
        - 预训练加微调的做法可以视为将大量参数进行分组，先对每组找到局部看起来比较好的设置，再基于局部较优的结果联合起来得到全局最优。
    - 权共享：weight sharing

- 一种节省训练开销的策略。即让一组神经元使用相同的连接权。该策略在CNN上发挥重要作用。
- CNN复合多个卷积层和采样层对输入信号进行加工，然后在连接层实现与输出目标之间的映射
- CNN使用BP算法训练，在训练中无论哪一层，同层的神经元都共享相同的连接权。
- CNN常将Sigmod激活函数替换为修正的线性函数$f(x) = max(0, x)$
- 示例：卷积神经网络用于手写数字识别

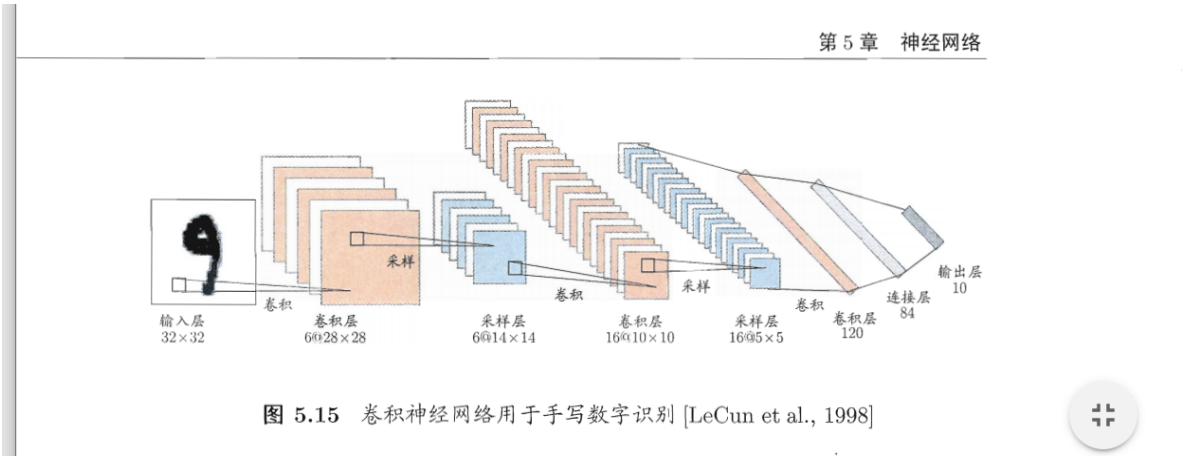


图 5.15　卷积神经网络用于手写数字识别 [LeCun et al., 1998]

深度学习的理解：

无论DBN or CNN，都是通过多层处理将初始的“底层”特征表示转化为高层特征表示后，用简单模型即可完成复杂的分类等学习任务。由此可以将深度学习理解为“特征学习”（feature learning）或者 表示学习（representation Learning）。特征学习通过机器学习技术自身来产生好特征，使得机器学习向“全自动数据分析”又前进了一步。