一、发展历史

- 第一次繁荣：感知机，但不能处理异或等线性不可分问题
- 第二次繁荣：Hopfield，BP可以优化多层感知机模型，多层感知机可以解决线性不可分。SVM使统计学习兴起，给大家信心什么时候能做好

> 神经网络需要调参技巧才能变好，而且究竟为什么很好很难解释基于经验
> SVM基于统计学习的理论，理论完备

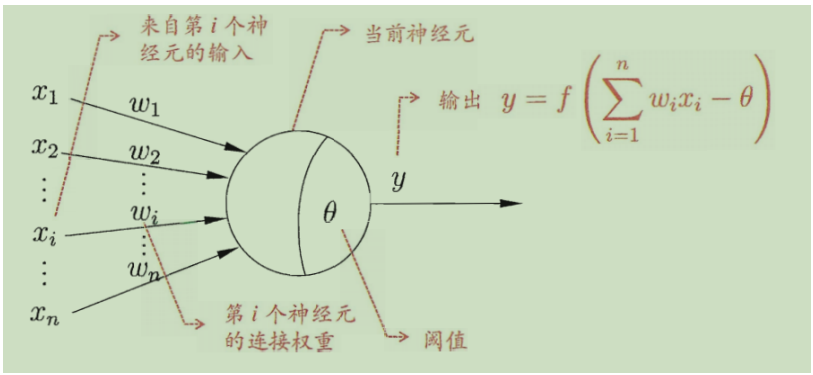
- 第三次繁荣：深度学习，有效训练深层神经网络，怎么优化

激活函数使神经网络变成非线性模型

无激活函数时，单隐层神经网络和线性模型区别不大

二、神经网络基础

1、M-P神经元模型



超过阈值输出1

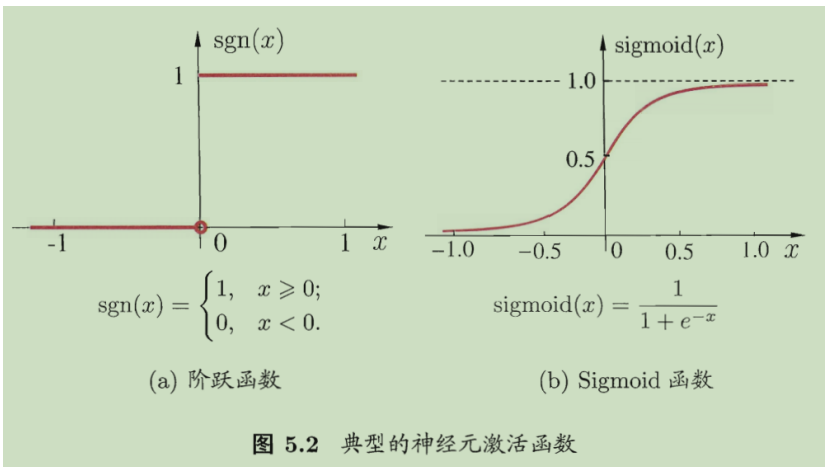


图 5.2 典型的神经元激活函数

2、感知机

> 两层神经元

- 支持与或非（线性可分问题，线性超平面可划分）
- 不支持异或

权重调整

$$w_i \leftarrow w_i + \Delta w_i \ ,$$

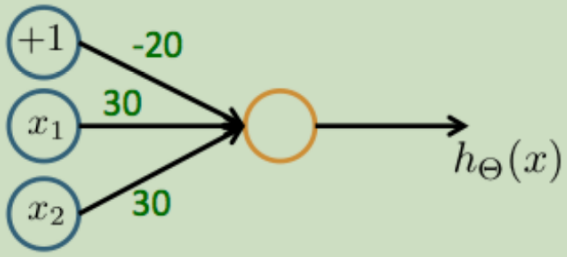
$$\Delta w_i = \eta(y - \hat{y})x_i \ ,$$

η学习率，一般0.1

> 学习率控制下降速度，太大可能震荡，太小会慢

- 统一为权重学习：阈值θ看作固定输入为-1的哑节点（dummy node）所对应的链接权重ω_{n+1}

69.考虑以下两个二值输入$x_1, x_2 \in \{0,1\}$和输出$h_\Theta(x)$的神经网络。它（近似）计算了下列哪一个逻辑函数？



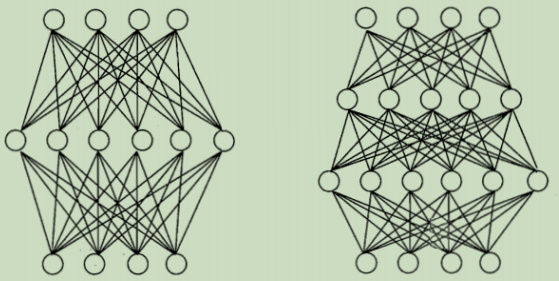
A. OR

3、多层前馈神经网络

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

> 有隐层就是多层

- 前馈：网络拓扑结构上不存在环或回路，但信号可以向后传
- 同层不连，跨层不连，邻层全连



- 学到的东西是连接权和阈值

4、标准BP误差逆传播算法

> 每次仅针对一个训练样例更新连接权和阈值

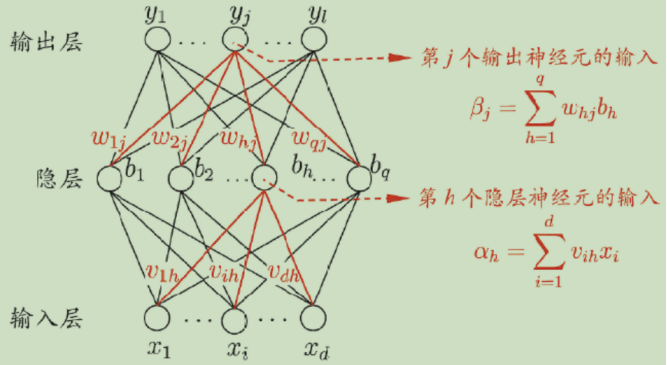
> 高效计算每一个参数的梯度，巧妙在于计算所需在计算损失时已算过

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



- 共需学习（d+l+1）q+l个参数

最小化均方误差解出权重

- Sigmoid输出预测：用输入-阈值

对训练例 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, 假定神经网络的输出为 $\hat{\boldsymbol{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$, 即

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

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

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

- Softmax输出

2. 当 $l > 1$, 网络的预测结果为 $\hat{\boldsymbol{y}} \in \mathbb{R}^l$, 其中 $\hat{y}_i$ 表示输入被预测为第 i 类的概率. 对于第 i 类的样本, 其标记 $\boldsymbol{y} \in \{0, 1\}^l$, 有 $y_i = 1$, $y_j = 0, j \neq i$. 对于一个训练样本 $(\boldsymbol{x}, \boldsymbol{y})$, 交叉熵损失函数 $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$ 的定义如下

$$\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{j=1}^{l} y_j \log \hat{y}_j \tag{2}$$

在多分类问题中, 一般使用 Softmax 层作为输出, Softmax 层的计算公式如下

$$\hat{y}_j = \frac{e^{\beta_j}}{\sum_{k=1}^{l} e^{\beta_k}} \tag{3}$$

易见 Softmax 函数输出的 $\hat{\boldsymbol{y}}$ 符合 $\sum_{j=1}^{l} \hat{y}_j = 1$, 所以可以直接作为每个类别的概率. Softmax 输出一般配合交叉熵 (Cross Entropy) 损失函数使用, 请计算 $\frac{\partial \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \beta_i}$,

迭代

- 基于梯度下降策略，以目标的负梯度方向对参数进行调整

$$v \leftarrow v + \Delta v .$$

举例--链式法则求梯度

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

注意到 w_{hj} 先影响到第 j 个输出层神经元的输入值 β_j, 再影响到其输出值 $\hat{y}_j^k$, 然后影响到 E_k, 有

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} . \tag{5.7}$$

根据 β_j 的定义, 显然有

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h . \tag{5.8}$$

图 5.2 中的 Sigmoid 函数有一个很好的性质:

$$f'(x) = f(x)(1 - f(x)) , \tag{5.9}$$

于是根据式(5.4)和(5.3), 有

$$\begin{aligned} g_j &= -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \\ &= -(\hat{y}_j^k - y_j^k) f'(\beta_j - \theta_j) \\ &= \hat{y}_j^k (1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k) . \end{aligned} \tag{5.10}$$

将式(5.10)和(5.8)代入式(5.7), 再代入式(5.6), 就得到了BP 算法中关于 w_{hj} 的更新公式

$$\Delta w_{hj} = \eta g_j b_h . \tag{5.11}$$

结论

$\Delta w_{hj} = \eta g_j b_h .$

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

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

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

$$
\begin{aligned}
e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \\
&= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j=1}^{l} w_{hj} g_j f'(\alpha_h - \gamma_h) \\
&= b_h(1-b_h) \sum_{j=1}^{l} w_{hj} g_j \ .
\end{aligned}
$$

5、BP算法分析

1、算法

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

过程:

1: 在$(0,1)$范围内随机初始化网络中所有连接权和阈值
2: **repeat**
3: **for all** $(\boldsymbol{x}_k, \boldsymbol{y}_k) \in D$ **do**
4: 根据当前参数和式(5.3) 计算当前样本的输出 $\hat{\boldsymbol{y}}_k$;
5: 根据式(5.10) 计算输出层神经元的梯度项 g_j;
6: 根据式(5.15) 计算隐层神经元的梯度项 e_h;
7: 根据式(5.11)-(5.14) 更新连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h
8: **end for**
9: **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

- 计算输出层误差（第 4-5行）
- 将误差逆向传播至隐层神经元（第6行）
- 根据隐层神经元误差调整连接权和阈值（第7行）

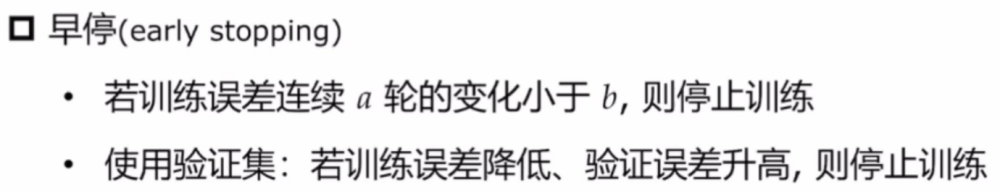
2、目标：min训练集上累计误差

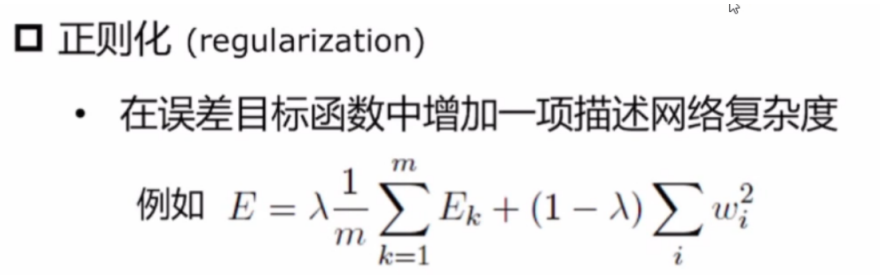
$$
E = \frac{1}{m} \sum_{k=1}^{m} E_k
$$

3、累积BP算法

- 直接对累积误差最小化，读取整个训练集D一遍后才对参数更新
- 标准BP算法每次更新只针对单个样例，参数更新非常频繁，对不同样例更新可能"抵消".
- 先用累计BP算法，缓慢后再用标准BP进一步优化

4、缓解过拟合





> 正则化刻画了结构风险
> 简单时早停和正则化效果类似

正则化改变了梯度更新的方法，让w先自身衰减一下

5、全局最小&局部最小

> 梯度只能得到局部最小

策略:

- 不同初始参数
- 随机梯度下降SGD的随机扰动

> 即便陷入局部极小点，计算出的梯度仍可能不是0

- 模拟退火：每一步都有一定概率接受比当前解更差的结果

三、拓展

1、RBF径向基函数

> Radial Basis Function
> 单隐层

- 定义特殊神经元

使用径向基函数作为隐层神经元激活函数

例如高斯径向基函数 $\rho(\boldsymbol{x}, \boldsymbol{c}_i) = e^{-\beta_i \|\boldsymbol{x}-\boldsymbol{c}_i\|^2}$

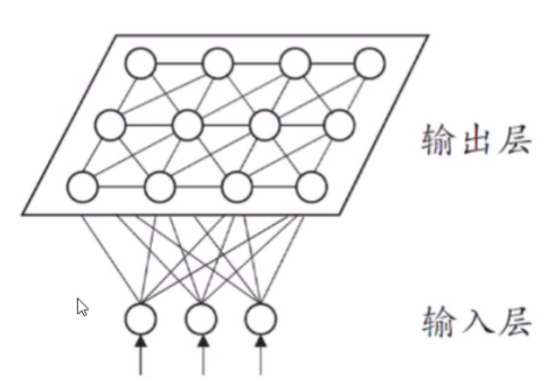
> 计算样本和ci的距离，指数加权，类似高斯核

输出层是隐层神经元输出的线性组合

$$\varphi(\boldsymbol{x}) = \sum_{i=1}^{q} w_i \rho(\boldsymbol{x}, \boldsymbol{c}_i)$$

- 先通过随机采样或聚类确定ci，用于确认神经元中心
- BP算法计算剩下参数

2、SOM自组织特征映射



> Self-Organizing feature Map
>
> 竞争型，无监督

- 只有输入层和网状输出层

> 输出层通关远近刻画关系远近
>
> 输入层表示不同样本，而不是不同维度，所以需要指派

作用：

- 只能降2维：但保持拓扑结构，适合可视化
- 聚类：用一个样本代表一批样本

目标：

- 每个神经元有一个权向量为每个输出层神经元找到合适的权向量来保持拓扑

> 高维中类似的样本会映射到相同低维神经元上

竞争学习

> 胜者通吃：获胜的神经元能代表一批样本

> 所以可以用少量神经元表示大量的拓扑

- 获胜神经元权向量向当前输入样本移动，实现更新

3、级联相关网络CC

> Cascade-Correlation

构造性神经网络：网络结构也是学习目标，同时优化神经网络结构，不断加入隐层

缺点：太动态，难以稳定，没有得到广泛应用

4、Elman(RNN)

> Recurrent NN & Recursive NN

- 神经元的输出可以作为同层神经元的输入
- t时刻输出由t时刻输入和t-1时刻网络状态共同决定

Elman 网络是最常用的递归神经网络之一

- 结构与前馈神经网络很相似，但隐层神经元的输出被反馈回来
- 使用推广的BP算法训练

目前在自然语言处理等领域常用的 LSTM 网络，是一种复杂得多的递归神经网络

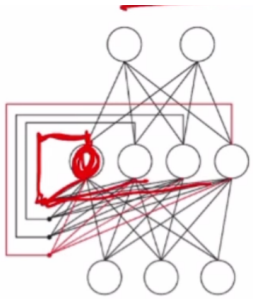


图 5.13 Elman 网络结构

四、深度学习的兴起

1、CNN卷积神经网络

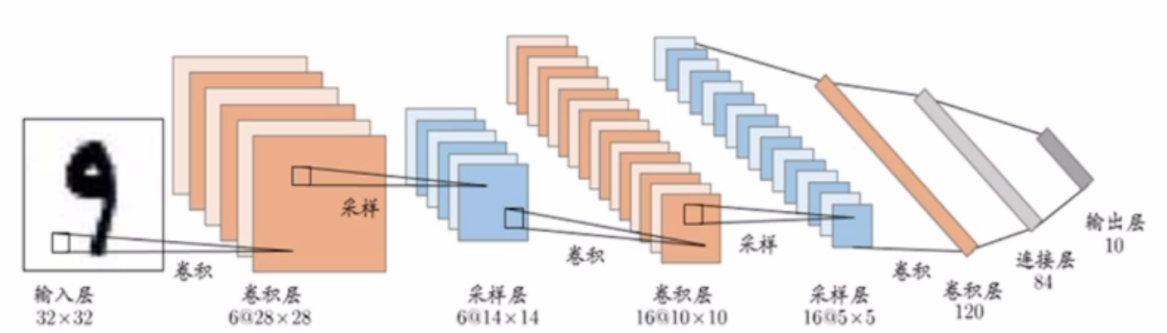
> Convolutional NN
>
> 卷积是信号层面的概念（数字信号处理）

多层函数嵌套，受到生物启发，但没有受到指导，

善于处理序列化数据

模板不需要人工设计，可以自动生成

多层前馈网络处理图像需要拉成向量



- 用多个模板扫描图片生成卷积层，某个局部匹配值越大说明和模板越像、越激活
- 扫描过后的卷积层面积会比输入层稍微小一点，因为模板也是有面积的
- 采样层：删减卷积层，选取少量数据代表大量数据，比如对几个元素算均值、最值
- 不断卷积--采样，最后全连接层投影到一维上

2、重要诀窍（tricks）

> trick重要性甚至高于方法
> 但有时候不知道到底哪个诀窍起了作用，难以解释
> 实践才能发现暗藏的规律，有效的调参实际上空间不大

预训练+微调

> 早期经验
> 预训练需要超大规模数据，在新任务的少量数据上微调

- 监督学习逐层训练，每次训练一层隐结点，引出一层目标函数，使监督信号不随层数增加而减弱
- 最后再对全网络微调

权共享

> 权重很多是模型优化空间大

- 一组神经元使用相同的连接权值

Dropout

> 深度神经网络能力强不怕难

- 任务变难可以避免过拟合，可以学到关键信息
- 训练阶段加随机噪声（置0）可以变难
- 变相集成学习

ReLU

> rectified Linear Units

- max(0,x)
- 激活函数变成修正线性函数,更适合深度学习的激活函数

- sigmoid缺点是两边太平滑，导数小导致多层时导数乘积更小（梯度消失问题）
- 进一步改进成leaky ReLU

交叉熵

- 交叉熵（对率回归）代替均方误差
- log对深层有好处，抵消一些数值的影响（如softmax中的exp）