

神经网络

翟婷婷

扬州大学
信息工程（人工智能）学院
zh tt@yzu.edu.cn

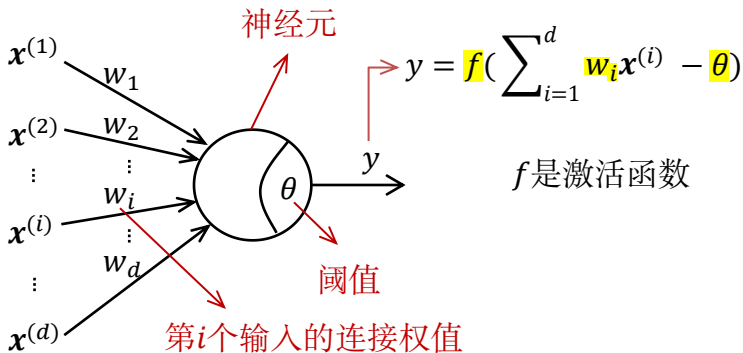
2023春

课程目标

- 理解人工神经网络的基本概念和用于模式识别的基本原理；
- 掌握BP算法的原理，会推导BP算法的更新公式，并能够编程实现BP算法，理解BP算法的局限性；
- 了解深度学习的基本思想和深度神经网络的特点；
- 了解卷积神经网络的基本工作原理，会利用已有平台编程实现卷积神经网络，并用于解决模式识别问题。

神经网络简介—神经元模型

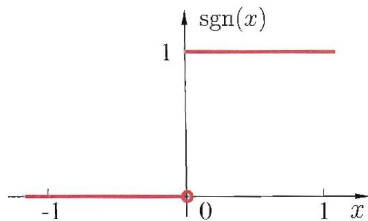
- 神经网络是很多的**神经元**模型按照一定的层次结构连接起来所构成的。



图：单个神经元模型（M-P神经元模型）

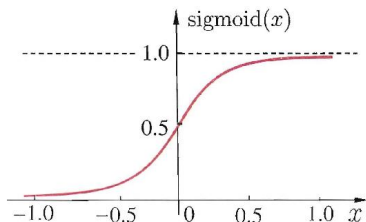
神经网络简介—激活函数

- 阶跃函数：它将输入值映射为0或1，0表示抑制神经元，1表示激活神经元。阶跃函数不连续、不光滑。
- sigmoid函数：将 $(-\infty, +\infty)$ 范围内变化的值映射到 $(0,1)$ 范围内。该函数连续、光滑、单调。



$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0; \\ 0, & x < 0. \end{cases}$$

(a) 阶跃函数



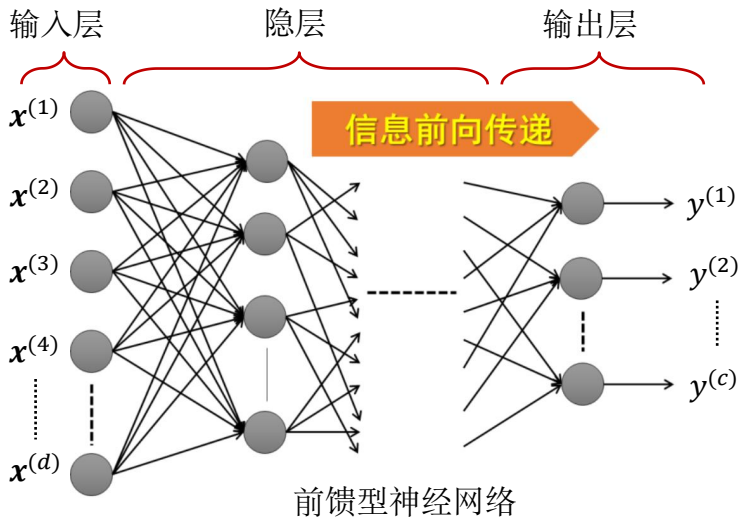
$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

(b) Sigmoid 函数

神经网络简介—网络结构

- 感知机是模拟**单个神经元**功能的一种数学模型，只能用于求解线性可分的问题。
- 单个神经元功能并不强大，但当神经元与神经元彼此连接起来构成一个复杂的网络时，就会发挥巨大能力。
- 根据神经元**连接方式**和**信息传递方向**，神经网络可以划分为**前馈型网络**和**反馈型网络**两大类。
- 在前馈型网络中，神经元有明显的层次结构，每层的神经元与下一层神经元**全互联**，同层的神经元之间不存在连接，跨层的神经元之间也不存在连接。神经元之间的连接不存在环或者回路。

神经网络简介—网络结构



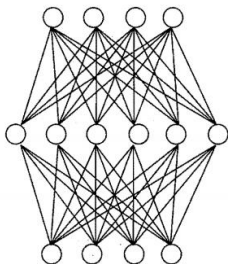
神经网络简介—网络结构

➤ 在前馈型神经网络中：

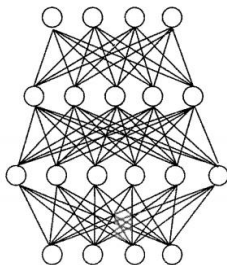
输入层神经元：仅接收外界输入，不对输入进行处理。

隐层和输出层神经元：是**拥有激活函数的功能神经元**，负责对输入进行加工，最终结果由输出层神经元输出。

只要包含隐层的网络，就可称为多层网络。



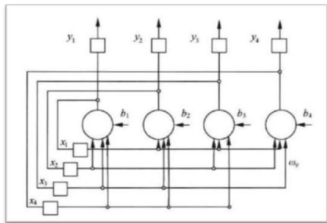
(a) 单隐层前馈网络



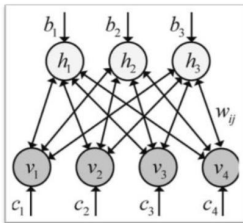
(b) 双隐层前馈网络

神经网络简介—网络结构

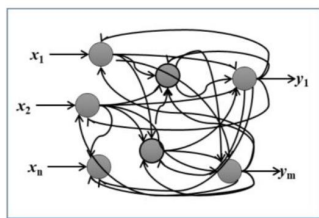
- 在反馈型神经网络中，存在从靠近输出端的神经元向靠近输入端的神经元的连接，因此，网络中存在环或回路，也可以说网络中存在信息的反馈。



Hopfield网络



受限玻尔兹曼机



任意反馈连接

- 反馈型神经网络的动态行为比较复杂，研究起来比较困难。因此，本课程重点学习前馈型网络。

神经网络简介—发展历史


➤ 1940年代~萌芽期:

M-P神经元模型(1943), Hebb学习规则

➤ 1958左右—1969左右~繁荣期:

感知机(1958), Adaline(1960), ...

➤ 1969年: Minsky & Papert 《Perceptron》




冰河期:
近15年

➤ 1985左右-1995左右~繁荣期:

Hopfield(1983), BP(1986),...

➤ 1995左右: SVM及统计学习兴起

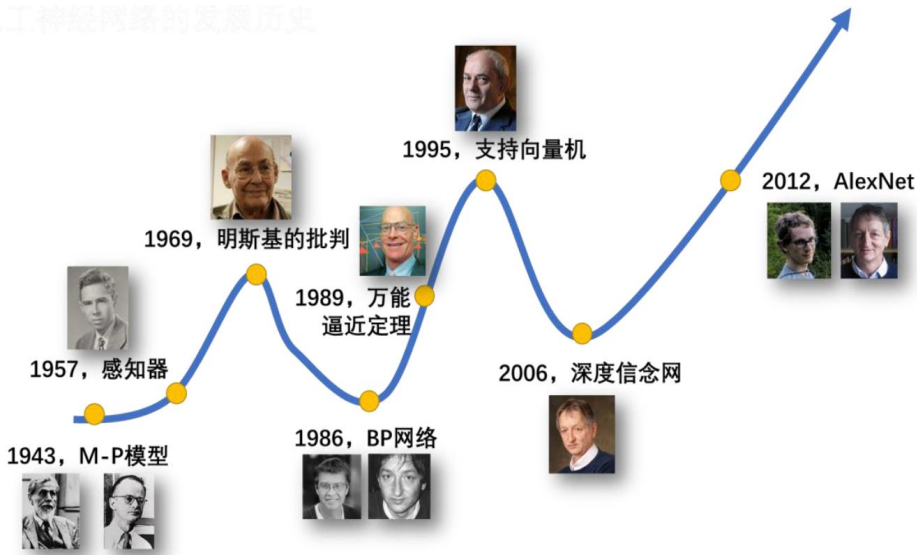


沉寂期:
近15年

➤ 2010至今~繁荣期: 深度学习

神经网络简介—发展历史

神经网络的发展历史

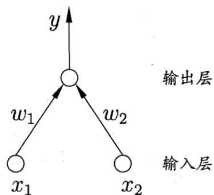


神经网络简介—训练/学习

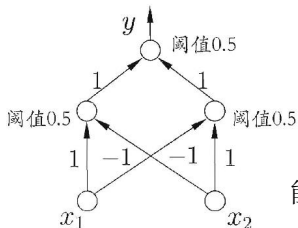
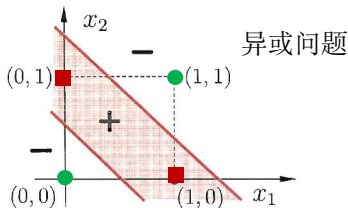
- 在确定使用神经网络作为模式识别的算法后，需要
 - ✿ **设计网络结构**：包括网络层数、每层神经元的个数、神经元之间的连接关系，神经元采用的激活函数形式。
 - ✿ **训练网络参数**：使用训练数据集学习得到神经元之间的连接权值和功能神经元的阈值。
- 神经网络的训练/学习过程，就是根据训练数据来调整**神经元之间的"连接权"**以及**每个功能神经元的阈值**。
换言之，神经网络"学"到的知识蕴涵在**连接权与阈值**中。

前馈神经网络

- 感知机实际上是一个无隐层的前馈神经网络，无法解决异或问题，而两层感知机就可以解决异或问题。



具有两个输入的单层感知机网络



能解决异或问题的两层感知机

前馈神经网络

- 多层前馈网络有强大的表示能力:

[Hornik et al., 1989] 证明, 如果一个多层前馈神经网络的隐层能够包含**足够多**的神经元, 那么该网络就能以**任意精度**逼近任意复杂度的连续函数。

然而, 如何设置隐层神经元的数目仍是个尚未解决的问题, 实际应用中通常靠"试错法" (trial-by-error) 调整。

- 多层网络的学习能力比单层感知机强得多, 要训练多层网络, 简单的感知机的学习规则显然不够, 需要更强大的学习算法。
- 误差反向传播(error Back Propagation, 简称 BP) 算法是最成功、最常用的神经网络学习/训练算法。

前馈神经网络的训练—BP算法

- 给定一个训练数据集 $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$, 其中 $\mathbf{x}_t \in \mathbb{R}^d$, $\mathbf{y}_t \in \mathbb{R}^c$ 。

注意 \mathbf{y}_t 是第 t 个样本的**类标向量**，如果一个样本属于第 k 类，则 \mathbf{y}_t 的第 k 个元素为1，其余为0：

$$\mathbf{y}_t = \begin{pmatrix} 1 & 2 \cdots & k \cdots & c \\ 0 & 0 \cdots & 1 \cdots & 0 \end{pmatrix}$$

- 假定网络的结构给定，目标是利用训练数据集学习出网络的参数，即连接权值和阈值。

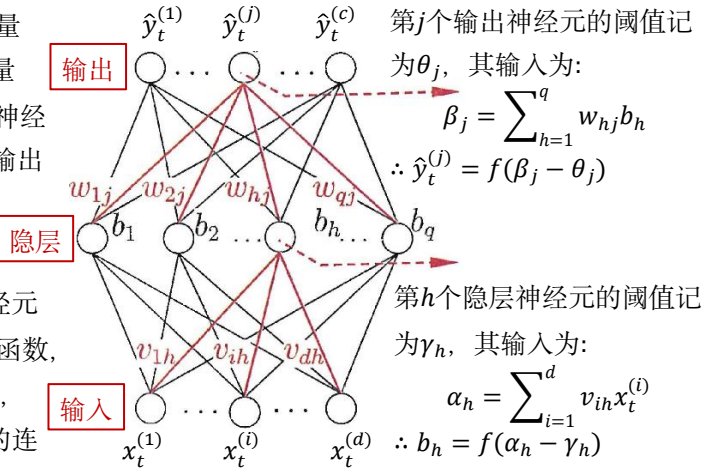
BP算法—网络结构和符号表示

输入: d 维特征向量

输出: c 维类标向量

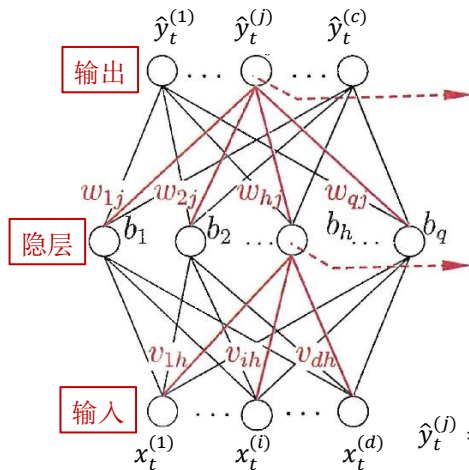
隐层: 假定有 q 个神经元, 各神经元的输出为: $b_1, b_2 \dots b_q$ 。

假定每个功能神经元采用sigmoid激活函数, 激活函数用 f 表示, 各层神经元之间的连接权如图所示。



该网络需要学习的参数有多少个? $d * q + q + q * c + c$

BP算法—网络结构和符号表示



输出层阈值: $\theta_1, \theta_2 \cdots \theta_c$

连接权:
$$\begin{bmatrix} w_{11} & w_{12} \cdots & w_{1c} \\ w_{21} & w_{22} \cdots & w_{2c} \\ \vdots & \vdots & \vdots \\ w_{q1} & w_{q2} \cdots & w_{qc} \end{bmatrix}$$

隐层阈值: $\gamma_1, \gamma_2 \cdots \gamma_q$

连接权:
$$\begin{bmatrix} v_{11} & v_{12} \cdots & v_{1q} \\ v_{21} & v_{22} \cdots & v_{2q} \\ \vdots & \vdots & \vdots \\ v_{d1} & v_{d2} \cdots & v_{dq} \end{bmatrix}$$

$$\hat{y}_t^{(j)} = f(\beta_j - \theta_j), \quad \beta_j = \sum_{h=1}^q w_{hj} b_h$$

$$b_h = f(\alpha_h - \gamma_h), \quad \alpha_h = \sum_{i=1}^d v_{ih} x_t^{(i)}$$

BP算法原理

- BP算法的学习目标是：找到一组网络参数，最小化在全体训练样本上的训练误差 $\frac{1}{2} \sum_{t=1}^N \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2$ ，其中， $\hat{\mathbf{y}}_t$ 是网络对 \mathbf{x}_t 的预测。BP算法采用“**随机梯度下降**”(**SGD**) 算法最小化在当前样本上的训练误差。
- 对第 t 个训练样本 $(\mathbf{x}_t, \mathbf{y}_t)$ ，假定神经网络的输出为

$$\hat{\mathbf{y}}_t = (\hat{y}_t^{(1)}, \hat{y}_t^{(2)} \dots \hat{y}_t^{(c)})$$

而理想的输出则为 $\mathbf{y}_t = (y_t^{(1)}, y_t^{(2)} \dots y_t^{(c)})$ ，则**当前误差**：

$$E_t = \frac{1}{2} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2 = \frac{1}{2} \sum_{j=1}^c \left(\hat{y}_t^{(j)} - y_t^{(j)} \right)^2$$

- 根据SGD，每次迭代中按照 E_t 的负梯度方向调整网络参数 u ：

$$u \leftarrow u - \eta \frac{\partial E_t}{\partial u}, \quad \text{其中 } \eta \text{ 是学习步长}$$

BP算法的推导

➤ 对输出层第 j 个神经元的阈值: $\theta_j \rightarrow \hat{y}_t^{(j)} \rightarrow E_t$

由**求导的链式法则**得:

$$\frac{\partial E_t}{\partial \theta_j} = \frac{\partial E_t}{\partial \hat{y}_t^{(j)}} \cdot \frac{\partial \hat{y}_t^{(j)}}{\partial \theta_j} = (\hat{y}_t^{(j)} - y_t^{(j)}) \cdot \frac{\partial \hat{y}_t^{(j)}}{\partial \theta_j}$$

由于 $\hat{y}_t^{(j)} = f(\beta_j - \theta_j)$, $f(x) = \frac{1}{1+e^{-x}}$, 且

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

$$\therefore \frac{\partial \hat{y}_t^{(j)}}{\partial \theta_j} = -f(\beta_j - \theta_j)(1 - f(\beta_j - \theta_j)) = -\hat{y}_t^{(j)} (1 - \hat{y}_t^{(j)})$$

$$\therefore \frac{\partial E_t}{\partial \theta_j} = - \underbrace{(\hat{y}_t^{(j)} - y_t^{(j)}) \hat{y}_t^{(j)} (1 - \hat{y}_t^{(j)})}_{g_j} = -g_j$$

$$g_j$$

BP算法的推导

- 对隐层的第 h 个神经元到输出层的第 j 个神经元的连接权值: $w_{hj} \rightarrow \beta_j \rightarrow \hat{y}_t^{(j)} \rightarrow E_t$

由**求导的链式法则**得:

$$\frac{\partial E_t}{\partial w_{hj}} = \frac{\partial E_t}{\partial \hat{y}_t^{(j)}} \cdot \frac{\partial \hat{y}_t^{(j)}}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

$$\hat{y}_t^{(j)} = f(\beta_j - \theta_j), \quad \beta_j = \sum_{h=1}^q w_{hj} b_h, \quad \therefore \frac{\partial \beta_j}{\partial w_{hj}} = b_h$$

$$\frac{\partial \hat{y}_t^{(j)}}{\partial \beta_j} = f(\beta_j - \theta_j)(1 - f(\beta_j - \theta_j)) = \hat{y}_t^{(j)}(1 - \hat{y}_t^{(j)})$$

$$\therefore \frac{\partial E_t}{\partial w_{hj}} = (\hat{y}_t^{(j)} - y_t^{(j)}) \hat{y}_t^{(j)}(1 - \hat{y}_t^{(j)}) b_h = g_j b_h$$

BP算法的推导

➤ 对隐层的第 h 个神经元的阈值:

$$\gamma_h \rightarrow b_h \rightarrow \begin{cases} \beta_1 \rightarrow \hat{y}_t^{(1)} \\ \beta_2 \rightarrow \hat{y}_t^{(2)} \\ \vdots \\ \beta_c \rightarrow \hat{y}_t^{(c)} \end{cases} \rightarrow E_t$$

由求导的链式法则得:

$$\frac{\partial E_t}{\partial \gamma_h} = \sum_{j=1}^c \frac{\partial E_t}{\partial \hat{y}_t^{(j)}} \cdot \frac{\partial \hat{y}_t^{(j)}}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \gamma_h}$$
$$\frac{\partial E_t}{\partial \hat{y}_t^{(j)}} \cdot \frac{\partial \hat{y}_t^{(j)}}{\partial \beta_j} = g_j, \quad \frac{\partial \beta_j}{\partial b_h} = w_{hj} \quad (\beta_j = \sum_{h=1}^q w_{hj} b_h)$$

BP算法的推导

因为 $b_h = f(\alpha_h - \gamma_h)$

$$\therefore \frac{\partial b_h}{\partial \gamma_h} = -f(\alpha_h - \gamma_h)(1 - f(\alpha_h - \gamma_h)) = -b_h(1 - b_h)$$

代入以上各式:

$$\frac{\partial E_t}{\partial \gamma_h} = -b_h(1 - b_h) \underbrace{\sum_{j=1}^c g_j w_{hj}}_{e_h} = -e_h$$

➤ 注意: $\frac{\partial E_t}{\partial \gamma_h}$ 里包含了两次对激活函数求导。

BP算法的推导

➤ 对输入层第*i*个神经元到隐层第*h*个神经元的连接权值:

$$v_{ih} \rightarrow \alpha_h \rightarrow b_h \rightarrow \begin{cases} \beta_1 \rightarrow \hat{y}_t^{(1)} \\ \beta_2 \rightarrow \hat{y}_t^{(2)} \\ \vdots \\ \beta_c \rightarrow \hat{y}_t^{(c)} \end{cases} \rightarrow E_t$$

由求导的链式法则得:

$$\begin{aligned} \frac{\partial E_t}{\partial v_{ih}} &= \sum_{j=1}^c \frac{\partial E_t}{\partial \hat{y}_t^{(j)}} \cdot \frac{\partial \hat{y}_t^{(j)}}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{ih}} \\ \frac{\partial E_t}{\partial v_{ih}} &= b_h(1 - b_h)x_t^{(i)} \sum_{j=1}^c g_j w_{hj} = e_h x_t^{(i)} \\ (b_h &= f(\alpha_h - \gamma_h), \alpha_h = \sum_{i=1}^d v_{ih} x_t^{(i)}) \end{aligned}$$

BP算法的推导

- 在每次迭代中，各个网络参数的更新公式为：

$\forall j = 1, 2 \dots c$, 计算

$$g_j = (\hat{y}_t^{(j)} - y_t^{(j)}) \hat{y}_t^{(j)} (1 - \hat{y}_t^{(j)})$$

$\forall h = 1, 2 \dots q$, 计算

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

网络参数更新为：

$$\forall j = 1, 2 \dots c, \quad \theta_j \leftarrow \theta_j + \eta g_j$$

$$\forall j = 1, 2 \dots c, h = 1, 2 \dots q, \quad w_{hj} \leftarrow w_{hj} - \eta g_j b_h$$

$$\forall h = 1, 2 \dots q, \quad \gamma_h \leftarrow \gamma_h + \eta e_h$$

$$\forall i = 1, 2 \dots d, h = 1, 2 \dots q, \quad v_{ih} \leftarrow v_{ih} - \eta e_h x_t^{(i)}$$

BP算法的伪代码

输入: 训练集 $=\{(\mathbf{x}_t, \mathbf{y}_t)\}, t = 1, 2 \dots N$; 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 $(\mathbf{x}_t, \mathbf{y}_t)$ do

4: 输入 \mathbf{x}_t , 根据当前网络参数计算隐层输出 $b_1 \dots b_h, \dots b_q$

和输出层的输出 $\hat{\mathbf{y}}_t = (\hat{y}_t^{(1)}, \hat{y}_t^{(2)} \dots \hat{y}_t^{(c)})$

5: 计算输出层神经元的梯度项 g_j 和隐层的梯度项目 e_h

6: 更新所有的连接权 w_{hj}, v_{ih} 与阈值 θ_j, γ_h

7: end for

8: 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

BP算法总结

- 误差反向传播机制依赖于误差对每个网络参数的导数，这要求激活函数可导。BP网络对多层感知器网络的一个重要改进是用可导、有界的sigmoid函数作为每个神经元的激活函数，这使得误差反向传播成为可能。
- BP网络的输入层神经元数量是特征维度，输出层神经元数量是类别数，但是隐层的层数和神经元的数量却是可以任意设定的。如果隐层神经元数量太少，则网络能够调节的参数会太少，导致误差缩减得很慢;如果隐层神经元数量太多，又容易产生过拟合现象。
- BP 算法的本质是随机梯度下降法，容易陷入局部最优。

BP算法缺点

- 由于使用sigmoid激活函数，BP网络存在如下问题：

sigmoid函数 $f(x) = \frac{1}{1+e^{-x}}$ ，其导数

$$f'(x) = f(x) (1 - f(x)) = - \left(f(x) - \frac{1}{2} \right)^2 + \frac{1}{4} \leq \frac{1}{4}$$

对一个包含 p 层隐层的神经网络，对层数进行编号：

输入层为第0层，输出层为第 $p + 1$ 层

第 $p + 1$ 层神经元的梯度项中包含 $f'(x)$ ；

第 p 层神经元的梯度项中包含 $f'(x) \cdot f'(x)$ ；

第1层神经元的梯度项中包含 $f'(x) \cdot f'(x) \cdots f'(x)$

$p + 1$ 项

BP算法缺点

- 通过上面分析，看到BP网络存在如下问题是：

随着网络层数增多，靠近输入端的神经元的梯度项逐渐减小，直至消失(趋于0)，这种现象称为“**梯度消失**”，这使得无法对网络参数实现有效调节。因此，**BP网络适用于浅层网络的学习，不适合深层次网络的学习。**

- 缓解BP网络过拟合的一个办法：采用正则化的误差

$$E_t = \frac{1}{2} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2 + \lambda \sum_i u_i^2$$

其中， u_i 表示每一个连接权和阈值参数， λ 是正则化系数，需要调节。

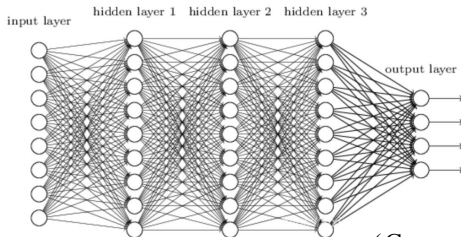
深度神经网络—深度学习的兴起

- 2006年：Hinton提出了深层网络训练中梯度消失问题的解决方案：无监督预训练对权值进行初始化+有监督训练微调。但是由于没有有效的实验验证，该论文并没有引起重视。
- 2011年，**ReLU激活函数**被提出，该激活函数能够有效的**抑制梯度消失**问题。同年，微软首次将DL应用在语音识别上，取得了重大突破。
- 2012年，Hinton课题组参加ImageNet图像识别比赛，使用构建的**CNN网络AlexNet**一举夺得冠军，且超过第二名（SVM方法）10多个百分点。从此，CNN吸引了计算机视觉、自然语言处理、语音识别等领域中众多研究者的关注，并且得到大型的IT企业的重视。

深度神经网络的特点

- 深度学习模型是**层数很多的大规模神经网络**，能实现非常复杂的非线性多分类映射关系。
- 深度学习的兴起与以 **GPU** 应用为标志的**大规模并行计算能力**的快速发展是密不可分的，因为只有能进行大规模并行计算的软硬件条件都具备了，才能支撑起网络规模大，神经元数量众多的深度神经网络的训练。
- 深度神经网络本质上是一个非常复杂的非线性模型，其 **VC 维**非常高，要降低结构风险，就一定要使用非常庞大的训练集，并确保训练集上的经验风险足够小。
深度模型的训练需要大量的训练样本作为支撑。
- 深度学习是“端到端”的学习，它将**特征学习**和**分类器学习**融合到了一个框架中，它能自动地学习特征的代表。

卷积神经网络(CNN)简介

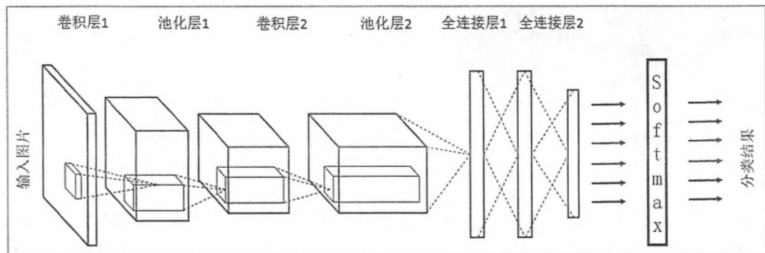


全连接的前馈神经网络

注意：全连接指的是相邻两层的神元之间全互联，但同层神元之间没有连接。

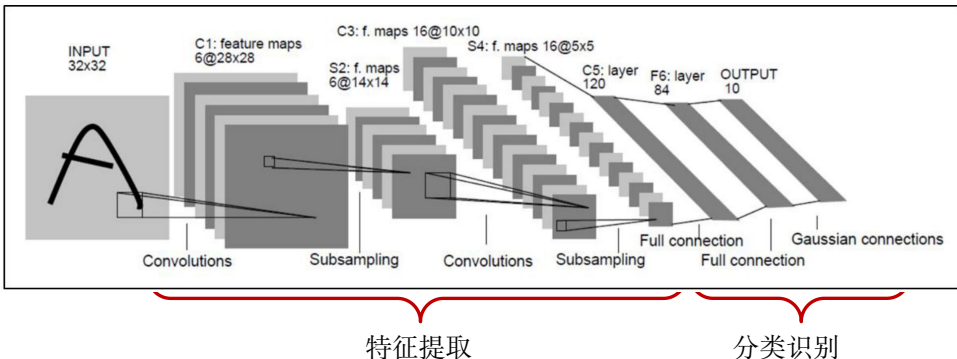
(Convolutional Neural Network, CNN)

用于图像分类的卷积神经网络图



卷积神经网络简介

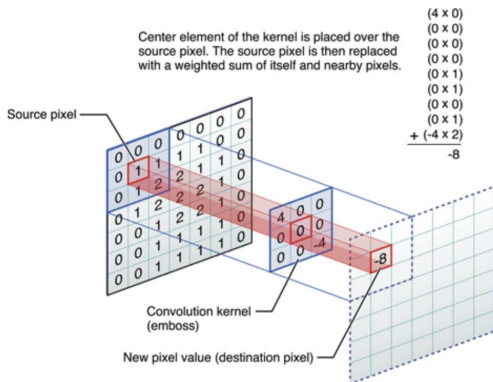
用于手写数字识别的卷积神经网络LeNet-5 [LeCun et al., 1998]



Le-net5的输入是一个二维图像，通过**多个卷积层**完成从低级到高级的**特征提取**，在卷积层之间，会通过**池化层**进行**特征降维**，最后通过一个**多层的全连接网络**构建一个分类器，以获得分类决策输出。

卷积神经网络—卷积层

- 卷积层使用多个不同的卷积核同时进行卷积操作，以实现对多种类型的特征进行提取。每个卷积核对原图像进行滑动遍历，就得到了一个新的二维图像，也就是提取出的对应特征的特征图。



卷积操作将输入图像中每一小块的像素进行更加深入地分析从而得到抽象程度更高的特征。

卷积运算

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2		

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3
2		

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3
2	3	

Convolved
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4
2	4	3
2	3	4

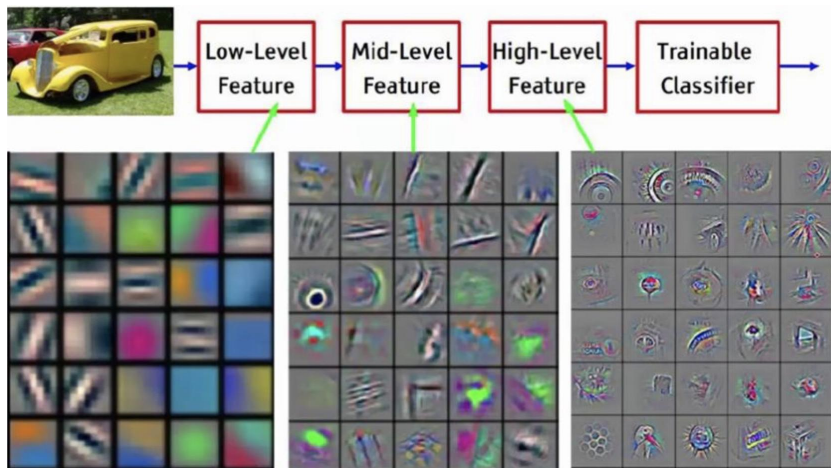
Convolved
Feature

特征图



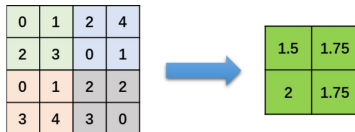
卷积运算

直观理解卷积操作:



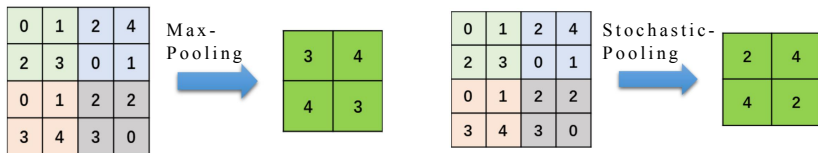
卷积神经网络—池化层

- 池化(pooling), 也称为下采样(subsampling), 是一种特征降维/图像压缩的技术。在二维图像中, 就是指将原始图像中一个局部区域内的多个像素值, 用1个像素值来代替的操作。
- 池化的区域通常取 2×2 , 并且无交叠地覆盖整个特征图。
- 常用的池化方法有:
 - ✿ 均值池化(Mean-Pooling): 将池化区域中的所有像素值用它们的均值来替代。可以较好地保留图像背景信息, 但是图像中的物体边缘会被钝化。



卷积神经网络—池化层

- ✿ 最大值池化(Max-Pooling): 将池化区域中的所有像素值用它们中的最大值来替代。能清晰地保留图像纹理信息，在物体轮廓等特征提取中很有效。



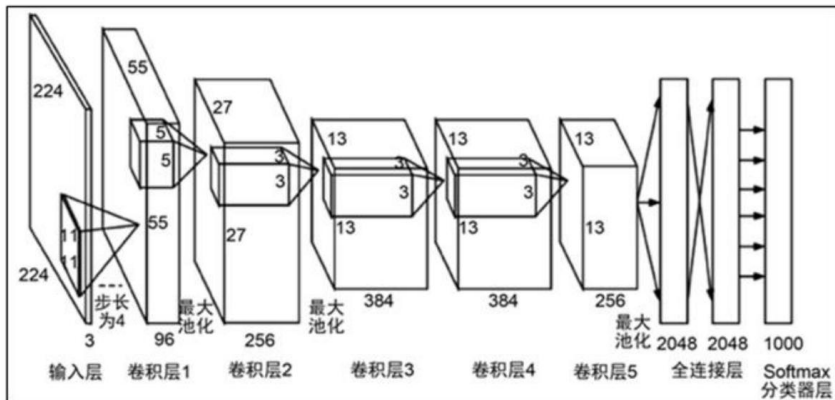
- ✿ 随机池化(Stochastic-Pooling): 根据池化区域中的像素值大小确定每个像素点被选中的概率，然后依概率随机选择一个像素点的像素值作为池化操作的结果。随机池化的效果比均值池化和最大值池化更好，但是计算量也更大。

卷积神经网络—训练/学习

- 卷积神经网络是典型的前馈型网络，因此，它的训练可以用BP算法来实现：
- ✿ 在正向状态更新中，从输入的图像开始，逐层对图像进行卷积操作和池化操作，不断提取高层特征，最后经全连接层和输出层，输出分类决策结果；
- ✿ 在反向误差传播时，首先根据输入图像的预期分类决策结果，就是图像的类别标签，计算出输出误差，再将输出误差逐层向前传播，来对每一层进行参数更新。

卷积神经网络—AlexNet

AlexNet网络结构



AlexNet的深度更深，它有**5个卷积层**，其中一部分卷积层后面接着有池化层；然后是**3个全连接层**，其中最后一层是 **softmax** 输出层，共有 **1000** 个节点，对应 ImageNet 图集中 **1000** 个图像分类。

卷积神经网络—AlexNet

- AlexNet所采用的重要技术包括：
 - ✿ ReLU激活函数：可以有效避免BP算法中的梯度消失；
 - ✿ dropout机制：在训练时，对全连接层随机的选择一部分神经元进行休眠，另外一些神经元参与网络的优化，即只有一部分特征会参与分类决策，可以大大降低网络过拟合的风险；
 - ✿ GPU加速：AlexNet使用2个GPU来完成网络的并行计算，大大提高了系统的性能。


深度学习的开源框架

- Caffe : 用C++编写的, 支持Python和MATLAB接口
<http://caffe.berkeleyvision.org>
- TensorFlow: Google公司开发, 支持多种语言
- PyTorch: Facebook公司开发, 使用Python
- MatConvNet: CNNs for MATLAB, MATLAB视觉工具箱
<https://www.vlfeat.org/matconvnet/>
-

小结

- 神经网络是很多的神经元模型按照一定的层次结构连接起来所构成的。
- 根据神经元连接方式和信息传递方向，神经网络可以划分为前馈型网络和反馈型网络两大类。
- 神经网络的训练/学习过程，就是在网络结构确定后，根据训练数据来调整神经元之间的"连接权"以及每个功能神经元的阈值。
- **【万能逼近定理】** 如果一个多层前馈神经网络的隐层能够包含足够多的神经元，那么该网络就能以任意精度逼近任意复杂度的连续函数。

小结

- 误差反向传播(error Back Propagation, 简称 BP) 算法是最成功、最常用的神经网络学习/训练算法, 它基于随机梯度下降法对网络参数进行更新, 主要用到的技术是求导的链式法则。
- 随着神经网络层数增多, 使用sigmoid激活函数的BP算法会出现“梯度消失”现象, 使得网络训练变得困难, 因此BP算法不能用于训练层数很深的深度神经网络。
- 随着深度神经网络的训练问题逐渐解决, 加之大数据技术和GPU大规模并行处理技术的发展, 促进深度学习的兴起, 并在产业界获得了广泛的应用。

小结

- 卷积神经网络(CNN)是一类具有代表性的深度神经网络，它因包含“卷积”操作而得名。
- 卷积神经网络往往包含多个卷积层完成从低级到高级的特征提取，在卷积层之间，会通过池化层进行特征降维，最后通过一个多层的全连接网络构建一个分类器，以获得分类决策输出。对CNN的训练仍可以使用BP算法。

难点