第一章

1.1 人工智能

人工智能概念：人工智能就是要让机器的行为看起来就像是人所表现出的智能行为一样。

主要领域：模拟人的感知、学习、认知。

范畴：机器感知（计算机视觉、语音信息处理）、学习（模式识别、 机器学习、强化学习）、语言（自然语言处理）、记忆（知识表示）、决策（规划、数据挖掘）等研究领域。

发展历史：1950，阿兰图灵，图灵测试。

1956，达特茅斯提出人工智能定义。

1956~20 世纪 70 年代，推理期。

1970~1986，知识期，出现专家系统。

1985~2005，学习期，机器学习、深度学习兴起。

1.3 表示学习

表示学习：如果有一种算法可以自动地学习出有效的特征，并提高最终机器学习模型的性能，那么这种学习就可以叫作表示学习。

语义鸿沟：问题是指输入数据的底层特征和高层语义信息之间的不一致性和差异性。

局部表示：一种表示颜色的方法是以不同名字来命名不同的颜色，这种表示方式叫作局部表示，也称为离散表示或符号表示。

分布式表示：另一种表示颜色的方法是用 RGB 值来表示颜色，不同颜色对应到 R、G、B 三维空间中一个点，这种表示方式叫作分布式表示。

表 1.1 局部表示和分布式表示示例

颜色	局部表示	分布式表示
琥珀色	$[1, 0, 0, 0]^T$	$[1.00, 0.75, 0.00]^T$
天蓝色	$[0, 1, 0, 0]^T$	$[0.00, 0.5, 1.00]^T$
中国红	$[0, 0, 1, 0]^T$	$[0.67, 0.22, 0.12]^T$
咖啡色	$[0, 0, 0, 1]^T$	$[0.44, 0.31\ 0.22]^T$

1.5 神经网络

人工神经网络是指由很多人工神经元构成的网络结构模型，这些人工神经元之间的连接强度是可学习的参数。

人类大脑是人体最复杂的器官，由神经元、神经胶质细胞、神经干细胞和血管组成。

典型的神经元结构大致可分为细胞体和细胞突起：细胞体、细胞凸起—树突，轴突。一个神经元可以被视为一种只有两种状态的细胞，有兴奋和抑制两种状态。

人脑有两种记忆：长期记忆和短期记忆。短期记忆转化为长期记忆的过程就称为凝固作用。

神经网络发展历史：　第一阶段为 1943 年～1969 年，MP 模型提出。

第二阶段为 1969 年～1983 年，冰河期，1974 发明反向传播算法。

第三阶段为 1983 年～1995 年，反向传播算法引起了复兴，但是梯度消失问题阻碍神经网络的进一步发展。

第四阶段为 1995 年～2006 年，支持向量机和其他更简单方法在机器学习领域的出现使神经网络流行度降低。

第五阶段为从 2006 年开始至今，深度学习使得神经网络再度崛起。

第四章 前馈神经网络

人工神经网络通过对人脑的神经元网络进行抽象，构建人工神经元，并按照一定拓扑结构来建立人工神经元之间的连接，来模拟生物神经网络。

人工神经元，简称神经元，是构成神经网络的基本单元，主要是模拟生物神经元的结构和特性，接受一组输入信号并产生输出。

4.1 常见激活函数

Sigmoid 函数：指一类 S 型曲线函数，为两端饱和函数。常见的 sigmoid 型函数有 Logistic 函数和 Tanh 函数。

Logistic 函数　Logistic 函数定义为

$$\sigma(x) = \frac{1}{1+\exp(-x)}.$$

Tanh 函数　Tanh 函数也是一种 Sigmoid 型函数. 其定义为

$$\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}.$$

Tanh 函数可以看作放大并平移的 Logistic 函数，其值域是 $(-1, 1)$.

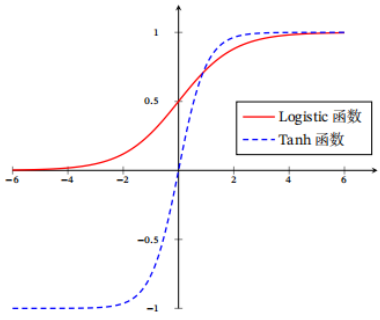


图 4.2　Logistic 函数和 Tanh 函数

非零中心化的输出会使得其后一层的神经元的输入发生偏置偏移（Bias Shift），并进一步使得梯度下降的收敛速度变慢。

$$\text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max(0, x).$$

ReLU 函数：实际上是一个斜坡函数，此外还有 ELU 函数与 Softplus 函数。

优点：具有生物学合理性，单侧抑制，宽兴奋边界。相比于 Sigmoid 的两端饱和，ReLU 为左饱和函数，在一定程度上缓解了神经网络的梯度消失问题。

缺点：ReLU 函数的输出是非零中心化的，给后一层的神经网络引入偏置偏移，会影响梯度下降的效率。且还会有死亡 ReLU 问题，详情看书本 P83。

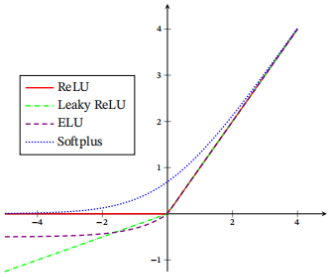


图 4.4 ReLU、Leaky ReLU、ELU 以及 Softplus 函数

死亡 ReLU 问题：如果参数发生不恰当更新，第一个隐藏层中的某一个 ReLU 神经元会永远不能激活，也就是这个神经元的自身参数的梯度永远是 0。（也可能发生在其他隐藏层）

Swish 函数：一种自门控激活函数，定义为 $\text{swish}(x) = x\sigma(\beta x),$

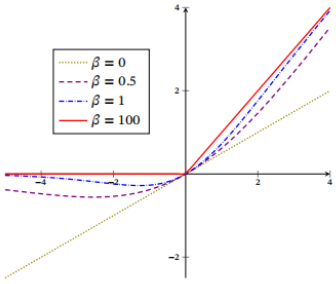


图 4.5 Swish 函数

GELU 函数：也是一种通过门控机制来调整其输出值的激活函数，和 Swish 函数比较类似.

$$\text{GELU}(x) = xP(X \leq x),$$

Maxout 单元：一种分段线性函数。Sigmoid 型函数、ReLU 等激活函数的输入是神经元的净输入 z，是一个标量。而 Maxout 单元的输入是上一层神经元的全部原始输出，是一个向量。

**4.3 前馈神经网络（构成、表达方式）

前馈神经网络是最早发明的简单人工神经网络，也经常被称为多层感知器。但它其实是由多

层的 Logistic 回归模型（连续的非线性函数）组成，而不是由多层的感知器（不连续的非线性函数）组成。

在前馈神经网络中，各神经元分别属于不同的层．每一层的神经元可以接收前一层神经元的信号，并产生信号输出到下一层．第 0 层称为输入层，最后一层称为输出层，其他中间层称为隐藏层．整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。

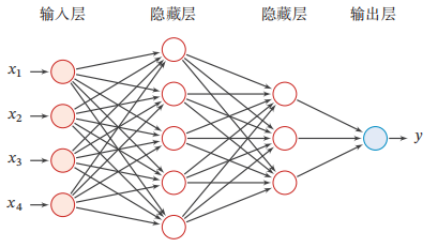


图 4.7　多层前馈神经网络

表 4.1　前馈神经网络的记号

记号	含义
L	神经网络的层数
M_l	第 l 层神经元的个数
$f_l(\cdot)$	第 l 层神经元的激活函数
$\boldsymbol{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第 $l-1$ 层到第 l 层的权重矩阵
$\boldsymbol{b}^{(l)} \in \mathbb{R}^{M_l}$	第 $l-1$ 层到第 l 层的偏置
$\boldsymbol{z}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的净输入（净活性值）
$\boldsymbol{a}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的输出（活性值）

令 $\boldsymbol{a}^{(0)} = \boldsymbol{x}$，前馈神经网络通过不断迭代下面公式进行信息传播：

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}, \tag{4.32}$$

$$\boldsymbol{a}^{(l)} = f_l(\boldsymbol{z}^{(l)}). \tag{4.33}$$

公式(4.32)和公式(4.33)也可以合并写为：

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)} f_{l-1}(\boldsymbol{z}^{(l-1)}) + \boldsymbol{b}^{(l)}, \tag{4.34}$$

或者

$$\boldsymbol{a}^{(l)} = f_l(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}). \tag{4.35}$$

这样，前馈神经网络可以通过逐层的信息传递，得到网络最后的输出。

****4.4 反向传播算法（认真看）

第 L 层的误差项可以通过第 L+1 层的误差项计算得到，这就是误差的反向传播。反向传播算法的含义是：第 L 层的一个神经元的误差项（或敏感性）是所有与该神经元相连的第 L+1 层 的

神经元的误差项的权重和。然后，再乘上该神经元激活函数的梯度。

通过计算三个偏导数

（1）计算偏导数 $\frac{\partial \boldsymbol{z}^{(l)}}{\partial w_{ij}^{(l)}}$　因 $\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$，偏导数

$$\begin{aligned}\frac{\partial \boldsymbol{z}^{(l)}}{\partial w_{ij}^{(l)}} &= \left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \cdots, \frac{\partial z_{M_l}^{(l)}}{\partial w_{ij}^{(l)}}\right] \\ &= \left[0, \cdots, \frac{\partial(\boldsymbol{w}_{i:}^{(l)}\boldsymbol{a}^{(l-1)} + b_i^{(l)})}{\partial w_{ij}^{(l)}}, \cdots, 0\right] \\ &= \left[0, \cdots, a_j^{(l-1)}, \cdots, 0\right] \\ &\triangleq \mathbb{I}_i(a_j^{(l-1)}) \quad \in \mathbb{R}^{1\times M_l},\end{aligned}$$

（2）计算偏导数 $\frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{b}^{(l)}}$　因为 $\boldsymbol{z}^{(l)}$ 和 $\boldsymbol{b}^{(l)}$ 的函数关系为 $\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$，因此偏导数

$$\frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{b}^{(l)}} = \boldsymbol{I}_{M_l} \quad \in \mathbb{R}^{M_l\times M_l}, \tag{4.55}$$

为 $M_l \times M_l$ 的单位矩阵.

（3）计算偏导数 $\frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l)}}$　偏导数 $\frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l)}}$ 表示第 l 层神经元对最终损失的影响，也反映了最终损失对第 l 层神经元的敏感程度，因此一般称为第 l 层神经元的误差项，用 $\delta^{(l)}$ 来表示.

$$\delta^{(l)} \triangleq \frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l)}} \quad \in \mathbb{R}^{M_l}. \tag{4.56}$$

根据链式法则，第 l 层的误差项为

$$\begin{aligned}\delta^{(l)} &\triangleq \frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l)}} \\ &= \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}} \cdot \frac{\partial \boldsymbol{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l+1)}} \\ &= \mathrm{diag}(f_l'(\boldsymbol{z}^{(l)})) \cdot (\boldsymbol{W}^{(l+1)})^{\mathrm{T}} \cdot \delta^{(l+1)} \\ &= f_l'(\boldsymbol{z}^{(l)}) \odot \left((\boldsymbol{W}^{(l+1)})^{\mathrm{T}}\delta^{(l+1)}\right) \quad \in \mathbb{R}^{M_l},\end{aligned}$$

然后通过三个偏导数更新公式

$$\frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \boldsymbol{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l)}}, \tag{4.49}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{b}^{(l)}} = \frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{b}^{(l)}} \frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}^{(l)}}. \tag{4.50}$$

得到

因此，$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$关于第$l$层权重$\boldsymbol{W}^{(l)}$的梯度为

$$\frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{W}^{(l)}} = \delta^{(l)} (\boldsymbol{a}^{(l-1)})^{\mathrm{T}} \quad \in \mathbb{R}^{M_l \times M_{l-1}}. \tag{4.68}$$

tps://nndl.github.io/

同理，$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})$关于第$l$层偏置$\boldsymbol{b}^{(l)}$的梯度为

$$\frac{\partial \mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{b}^{(l)}} = \delta^{(l)} \quad \in \mathbb{R}^{M_l}. \tag{4.69}$$

因此

使用误差反向传播算法的前馈神经网络训练过程可以分为以下三步：

（1） 前馈计算每一层的净输入$\boldsymbol{z}^{(l)}$和激活值$\boldsymbol{a}^{(l)}$，直到最后一层；

（2） 反向传播计算每一层的误差项$\delta^{(l)}$；

（3） 计算每一层参数的偏导数，并更新参数.

算法4.1给出使用反向传播算法的随机梯度下降训练过程.

算法 4.1　使用反向传播算法的随机梯度下降训练过程

输入: 训练集 $\mathcal{D}=\{(\boldsymbol{x}^{(n)},y^{(n)})\}_{n=1}^{N}$, 验证集 $\mathcal{V}$, 学习率 α, 正则化系数 λ, 网络层数 L, 神经元数量 $M_l, 1\leq l\leq L$.

随机初始化 $\boldsymbol{W},\boldsymbol{b}$;
repeat
　对训练集 $\mathcal{D}$ 中的样本随机重排序;
　for $n=1\cdots N$ **do**
　　从训练集 $\mathcal{D}$ 中选取样本 $(\boldsymbol{x}^{(n)},y^{(n)})$;
　　前馈计算每一层的净输入 $\boldsymbol{z}^{(l)}$ 和激活值 $\boldsymbol{a}^{(l)}$, 直到最后一层;
　　反向传播计算每一层的误差 $\delta^{(l)}$;　// 公式 (4.63)
　　// 计算每一层参数的导数
　　$\forall l,\ \frac{\partial \mathcal{L}(y^{(n)},\hat{y}^{(n)})}{\partial \boldsymbol{W}^{(l)}}=\delta^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathrm{T}}$;　// 公式 (4.68)
　　$\forall l,\ \frac{\partial \mathcal{L}(y^{(n)},\hat{y}^{(n)})}{\partial \boldsymbol{b}^{(l)}}=\delta^{(l)}$;　// 公式 (4.69)
　　// 更新参数
　　$\boldsymbol{W}^{(l)}\leftarrow \boldsymbol{W}^{(l)}-\alpha(\delta^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathrm{T}}+\lambda \boldsymbol{W}^{(l)})$;
　　$\boldsymbol{b}^{(l)}\leftarrow \boldsymbol{b}^{(l)}-\alpha\delta^{(l)}$;
　end
until 神经网络模型在验证集 $\mathcal{V}$ 上的错误率不再下降;
输出: $\boldsymbol{W},\boldsymbol{b}$

4.5 自动梯度计算

自动计算梯度的方法可以分为以下三类：数值微分、符号微分和自动微分。
数值微分是用数值方法来计算函数 f（x）的导数。数值微分容易实现，但找到一个合适的扰动△x 却十分困难。
符号微分是一种基于符号计算的自动求导方法，一般不需要代入具体的值。

自动微分是一种可以对一个（程序）函数进行计算导数的方法。原理是所有的数值计算可以分解为一些基本操作，然后利用链式法则来自动计算一个复合函数的梯度。
自动微分分为两种模式：前向模式和反向模式。
　前向模式是按计算图中计算方向的相同方向来递归地计算梯度。
　反向模式是按计算图中计算方向的相反方向来递归地计算梯度。

计算图按构建方式可以分为静态计算图和动态计算图。
静态计算图是在编译时构建计算图，计算图构建好之后在程序运行时不能改变，在构建时可以进行优化，并行能力强但灵活性较差。
动态计算图是在程序运行时动态构建，灵活性高但是不容易优化，难以并行计算。

4.6 优化问题

1、非凸优化问题
2、梯度消失问题，由于 Sigmoid 型函数的饱和性，饱和区的导数更是接近于 0。这样，误差经过每一层传递都会不断衰减．当网络层数很深时，梯度就会不停衰减，甚至消失，使得整个网络很难训练．这就是所谓的梯度消失问题。

第五章　卷积神经网络

卷积神经网络是一种具有局部连接、权重共享等特性的深层前馈神经网络。

用全连接前馈网络来处理图像时存在参数太多、局部不变性特征。因此过渡到了卷积神经网络，它是由卷积层(Convolution Layer)、汇聚层(Pooling Layer)和全连接层(Fully Connected Layer)交叉堆叠而成的前馈神经网络。卷积神经网络有三个结构上的特性：局部连接、权重共享以及汇聚。

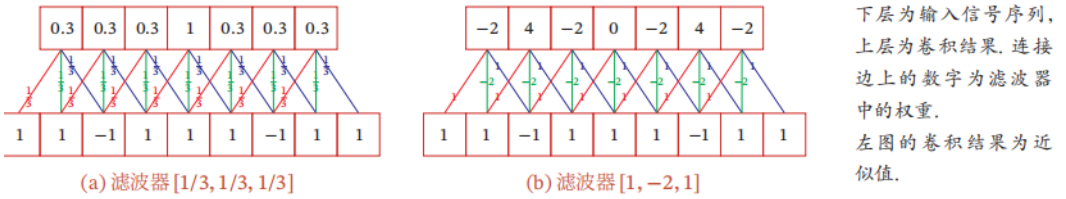
5.1 卷积

在信号处理或图像处理中，经常使用一维或二维卷积。

一维卷积：

当令滤波器 $\boldsymbol{w}=[1,-2,1]$ 时，可以近似实现对信号序列的二阶微分，即

$$x''(t)=x(t+1)+x(t-1)-2x(t). \tag{5.6}$$



(a) 滤波器 [1/3, 1/3, 1/3]　　(b) 滤波器 [1, −2, 1]

图 5.1　一维卷积示例

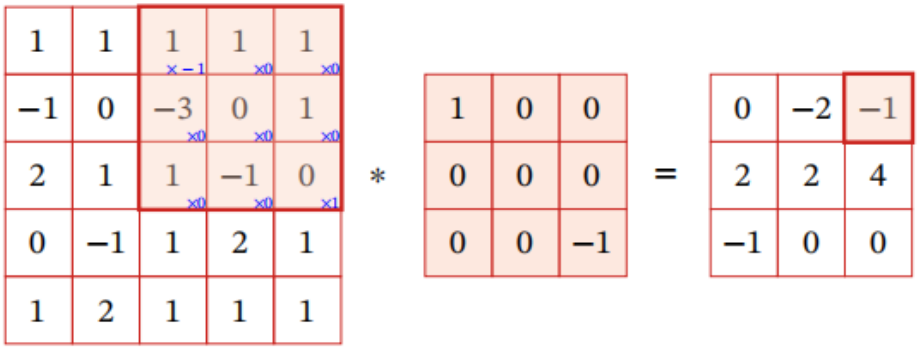


图 5.2　二维卷积示例

二维卷积：

均值滤波就是一种二维卷积

一幅图像经过卷积操作后得到结果称为特征映射。

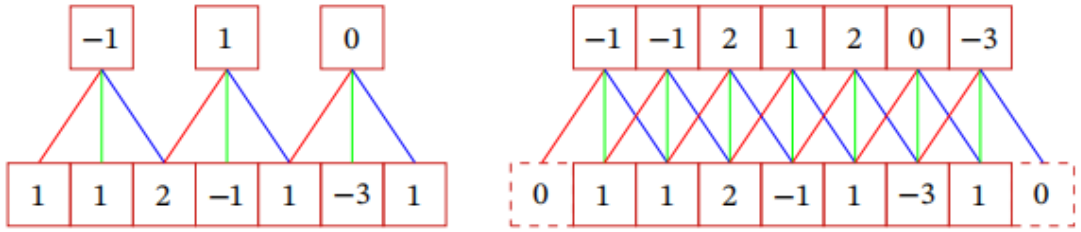
特征映射（Feature Map） 为一幅图像（或其他特征映射）在经过卷积提取到的特征，每个特征映射可以作为一类抽取的图像特征. 为了提高卷积网络的表示能力，可以在每一层使用多个不同的特征映射，以更好地表示图像的特征。

在输入为$\mathcal{X}\in\mathbb{R}^{M\times N\times D}$，输出为$\mathcal{Y}\in\mathbb{R}^{M'\times N'\times P}$的卷积层中，每一个输出特征映射都需要$D$个卷积核以及一个偏置. 假设每个卷积核的大小为$U\times V$，那么共需要$P\times D\times(U\times V)+P$个参数.

卷积的主要功能是在一个图像（或某种特征）上滑动一个卷积核（即滤波器），通过卷积操作得到一组新的特征。在计算卷积的过程中，需要进行卷积核翻转。（旋转 180 度）

互相关和卷积的区别仅仅在于卷积核是否进行翻转，因此互相关也可以称为不翻转卷积。

在卷积的标准定义基础上，还可以引入卷积核的滑动步长和零填充来增加卷积的多样性，可以更灵活地进行特征抽取。
步长是指卷积核在滑动时的时间间隔。
零填充是在输入向量两端进行补零。



(a) 步长 $S=2$　　(b) 零填充 $P=1$

图 5.4　卷积的步长和零填充（滤波器为 $[-1,0,1]$）

假设卷积层的输入神经元个数为 M，卷积大小为 K

窄卷积：步长 S= 1，两端不补零 P= 0，卷积后输出长度为 M-K+1

宽卷积：步长 S= 1，两端不补零 P= K-1，卷积后输出长度为 M+K-1

窄卷积：步长 S= 1，两端不补零 P= (K-1)/2，卷积后输出长度为 M

在输入为$\mathcal{X} \in \mathbb{R}^{M\times N\times D}$，输出为$\mathcal{Y} \in \mathbb{R}^{M'\times N'\times P}$的卷积层中，每一个输出特征映射都需要$D$个卷积核以及一个偏置．假设每个卷积核的大小为$U \times V$，那么共需要$P \times D \times (U \times V) + P$个参数．

5.2 卷积神经网络

卷积神经网络一般由卷积层、汇聚层和全连接层构成
局部连接：在卷积层中的每一个神经元都只和下一次中某个局部窗口内的神经元相连，构成一个局部连接网络。
权重共享：权重共享可以理解为一个卷积核只捕捉输入数据中的一种特定的局部特征。

在输入为$\mathcal{X} \in \mathbb{R}^{M\times N\times D}$，输出为$\mathcal{Y} \in \mathbb{R}^{M'\times N'\times P}$的卷积层中，每一个输出特征映射都需要$D$个卷积核以及一个偏置．假设每个卷积核的大小为$U \times V$，那么共需要$P \times D \times (U \times V) + P$个参数．

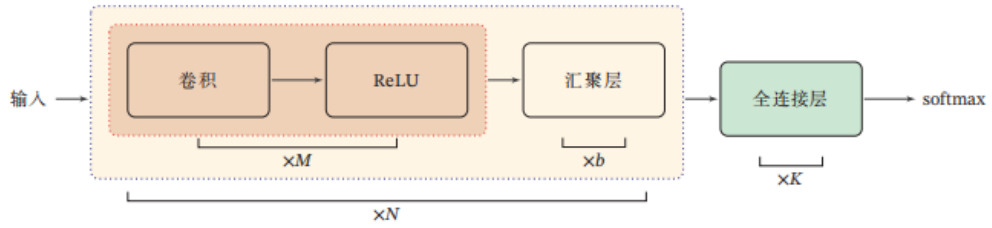
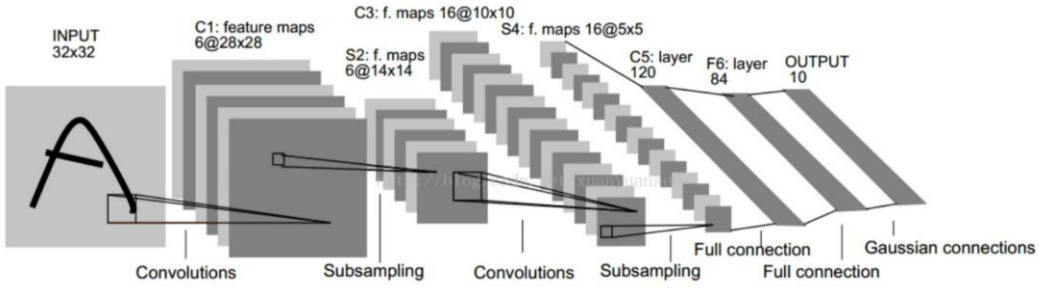


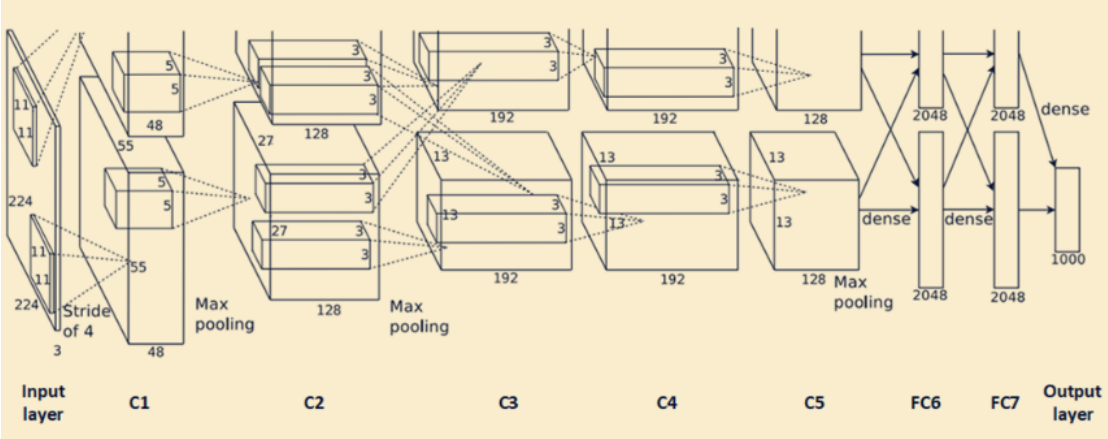
图 5.9　常用的卷积网络整体结构

5.4 常见的卷积神经网络

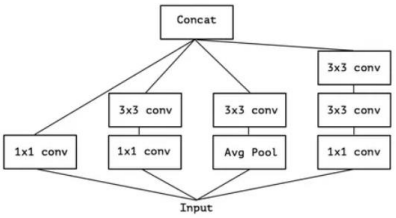
LeNet-5 ：共有七层，接受输入图像大小为 32*32=1024,输出对应 10 个类别的得分。基于 LeNet-5 的手写数字识别系统在上世纪 90 年代被美国很多银行使用。其输出层由 10 个径向基函数组成。



AlexNet：第一个现代深度卷积网络模型，2012 图像分类竞赛冠军，首次使用了很多现代深度卷积网络的技术方法，如使用 GPU 进行并行训练，采用了 ReLU 作为非线性激活函数，使用 Dropout 防止过拟合，使用数据增强来提高模型准确率等。其还在前两个汇聚层之后进行了局部相应归一化以增强模型的泛化能力。



Inception 网络：一个卷积层包含多个不同大小的卷积操作，称为 Inception 模块。2014 图像分类竞赛冠军。



残差网络 （残差连接）：通过给非线性的卷积层增加直连边的方式来提高信息的传播效率。

$$h(x) = \underbrace{x}_{\text{恒等函数}} + \underbrace{\big(h(x) - x\big)}_{\text{残差函数}}.$$

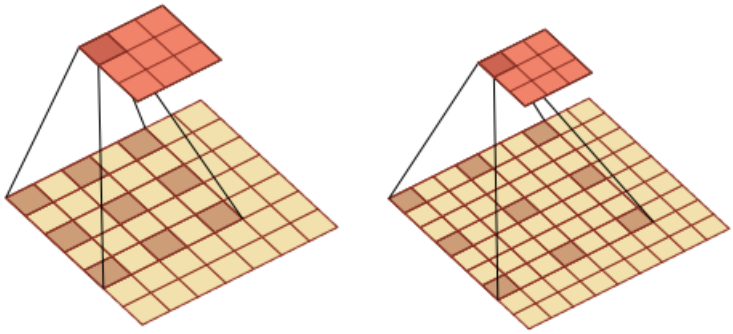
h（x）为目标函数。

5.5 空洞卷积

空洞卷积是一种不增加参数数量，同时增加输出单元感受野的一种方法，也称为膨胀卷积。空洞卷积通过给卷积核插入“空洞”来变相地增加其大小. 如果在卷积核的每两个元素之间插入 D－1 个空洞，卷积核的有效大小为$K' = K + (K-1) \times (D-1)$

其中 D 成为膨胀率，当 D= 1 时卷积核为普通的卷积核

图5.18给出了空洞卷积的示例.



(a) 膨胀率 $D = 2$　　(b) 膨胀率 $D = 3$

图 5.18　空洞卷积

。

第六章　循环神经网络

因为前馈神经网络中信息传递是单项的，虽然使得网络变得更容易学习，但在一定程度上也减弱了神经网络模型的能力，难以处理时序数据。而循环神经网络是一类具有短期记忆能力的能力更强的神经网络。不但可以接受其他神经元的信息，也可以接受自身的信息，形成具有环路的网络结构。

循环神经网络参数学习可以通过随时间反向传播算法来学习，但当输入序列比较长时，会存在梯度爆炸和消失问题，也称为长程依赖问题。最有效的改进方法是引入门控机制。

6.3 应用到机器学习

循环神经网络可以应用到很多不同类型的机器学习任务. 根据这些任务的特点可以分为以下几种模式：序列到类别模式、同步的序列到序列模式、异步的序列到序列模式。

序列到类别模式：主要用于序列数据的分类问题，输入为序列，输出为类别。比如在文本分类中，输入数据为单词的序列，输出为该文本的类别。

同步的序列到序列模式：主要用于序列标注任务，即每一时刻都有输入和输出，输入序列和输出序列的长度相同，比如在词性标注中，每一个单词都需要标注其对应的词性标签。

异步的序列到序列模式：也称为编码器-解码器模型，即输入序列和输出序列不需要有严格的对应关系，也不需要保持相同的长度．比如在机器翻译中，输入为源语言的单词序列，输出为目标语言的单词序列。

*反向传播算法名字：随时间反向传播算法、实时循环学习算法。两种算法都是基于梯度下降的算法，分别通过前向模式和反向模式应用链式法则来计算梯度。

6.5 长程依赖问题

循环神经网络在学习过程中的主要问题是由于梯度消失或爆炸问题，很难建模长时间间隔的状态之间的依赖关系。当时间间隔比较大时，若 $\gamma > 1$，梯度会变得很大，称为梯度爆炸问题；若 $\gamma < 1$，则会出现和深层前馈神经网络类似的梯度消失问题。

一般而言，循环网络的梯度爆炸问题比较容易解决，一般通过权重衰减或梯度截断来避免。
权重衰减 通过给参数增加 l1 或 l2 范数的正则化项来限制参数的取值范围，从而使得 $\gamma \leqslant 1$.
梯度截断 启发式方法，当梯度模大于一定阈值时，就将它截断成为一个较小的数。

***6.6 基于门控的循环神经网络

为改善长程依赖问题，引入门控机制，有选择性地加入新的信息，有选择地遗忘之前累积的信息。

长短期记忆网络（LSTM）：引入了一个新的内部状态进行线性的循环信息传递。引入了门控机制，三个门分别为遗忘门、输入门、输出门，是软门，取值在（0,1）之间。

LSTM 网络引入门控机制（Gating Mechanism）来控制信息传递的路径. 公式(6.51)和公式(6.52)中三个“门”分别为输入门 $\boldsymbol{i}_t$、遗忘门 $\boldsymbol{f}_t$ 和输出门 $\boldsymbol{o}_t$. 这三个门的作用为

（1）遗忘门 $\boldsymbol{f}_t$ 控制上一个时刻的内部状态 $\boldsymbol{c}_{t-1}$ 需要遗忘多少信息.

（2）输入门 $\boldsymbol{i}_t$ 控制当前时刻的候选状态 $\tilde{\boldsymbol{c}}_t$ 有多少信息需要保存.

（3）输出门 $\boldsymbol{o}_t$ 控制当前时刻的内部状态 $\boldsymbol{c}_t$ 有多少信息需要输出给外部状态 $\boldsymbol{h}_t$.

LSTM 网络中的“门”是一种“软”门，取值在 $(0,1)$ 之间，表示以一定的比例允许信息通过. 三个门的计算方式为：

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i), \tag{6.54}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f), \tag{6.55}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o), \tag{6.56}$$

其中 $\sigma(\cdot)$ 为 Logistic 函数，其输出区间为 $(0,1)$，$\boldsymbol{x}_t$ 为当前时刻的输入，$\boldsymbol{h}_{t-1}$ 为上一时刻的外部状态.

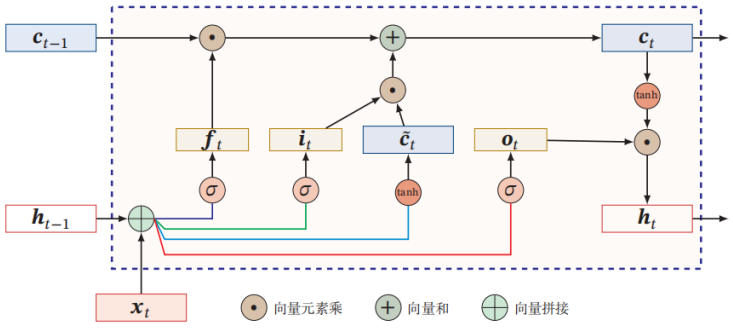


图 6.7　LSTM 网络的循环单元结构

（LSTM 还有各种变体：

无遗忘门的 LSTM：最早提出的 LSTM 网络是没有遗忘门的。当输入序列的长度非常大时，记忆单元的容量会饱和，从而大大降低 LSTM 模型的性能。

Peephole 连接：另外一个变体是三个门不但依赖于输入 Xt 和上一时刻的隐状态 Ht-1，也依赖于上一个时刻的记忆单元 Ct-1。

耦合输入门和遗忘门：LSTM 网络中的输入门和遗忘门有些互补关系，因此同时用两个门比较冗余。为了减少 LSTM 网络的计算复杂度，将这两门合并为一个门。）

门控循环单元网络（GRU）：和 LSTM 不同，GRU 不引入额外的记忆单元，但引入了一个更新门来控制当前状态需要从历史状态中保留多少信息以及从候选状态中接受多少新信息。

第七章网络优化与正则化

应用神经网络模型到机器学习时依然存在一些难点问题：

（1）优化问题：找到全局最优解比较困难，二阶优化方法计算代价很高通常无法使用，而一阶优化方法的训练效率通常比较低，还存在梯度爆炸或消失问题。

（2）泛化问题：神经网络容易在训练集上过拟合，需要通过正则化来改进网络的泛化能力。

优化算法有大体分为两类：1）调整学习率使优化更稳定。2）梯度估计修正，优化训练速度。

具体实现中，梯度下降法可以分为：批量梯度下降、随机梯度下降以及小批量梯度下降三种形式。

影响小批量梯度下降法的主要因素有：1）批量大小 K，2）学习率 α，3）梯度估计

批量大小越大，随机梯度的方差越小，引入的噪声也越小，训练也越稳定，因此可以设置较大的学习率.

批量大小较小时，需要设置较小的学习率，否则模型会不收敛.

学习率通常要随着批量大小的增大而相应地增大。

学习率是神经网络优化时的重要超参数，过大就不会收敛，过小则收敛速度太慢。从经验上看，学习率在一开始要保持大些来保证收敛速度，在收敛到最优点附近时要小些以避免来回振荡。

*表格

表 7.1　神经网络常用优化方法的汇总

类别		优化算法
学习率调整	固定衰减学习率	分段常数衰减、逆时衰减、(自然)指数衰减、余弦衰减
	周期性学习率	循环学习率、SGDR
	自适应学习率	AdaGrad、RMSprop、AdaDelta
梯度估计修正		动量法、Nesterov 加速梯度、梯度截断
综合方法		Adam≈动量法+RMSprop

循环学习率让学习率在一个区间内周期性地增大和缩小。
带热重启的随机梯度下降（SGDR）用热重启方式来替代学习率衰减。每次重启后模型参数不是从头开始优化而是从重启前的参数基础上基础优化。
AdaGrad 算法在每次迭代时自适应地调整每个参数的学习率而不是使用相同的学习率。
RMSprop 算法是一种自适应学习率的方法，有时可以避免 AdaGrad 算法中学习率不断单调下降以至于过早衰减的缺点。
AdaDelta 也是 AdaGrad 算法的一个改进，通过梯度平方的指数衰减移动平均来调整学习率。此外还引入了每次参数更新差值的平方的指数衰减权移动平均。

动量法用之前累积动量来替代真正的梯度，每次迭代的梯度可以看做加速度。
Nesterov 加速梯度是一种对动量法的改进。
梯度截断就是当梯度的模大于一定阈值时，就对梯度进行截断。

Adam 算法可以看做动量法和 RMSprop 算法的结合，使用动量作为参数更新方向，还可以自适应调整学习率。

**7.3 参数的初始化（一节看完）

参数初始化的方法通常有以下三种：
（1）预训练初始化：一个已经在大规模数据上训练过的模型可以提供一个好的参数初始值。
（2）随机初始化：对每个参数都随机初始化，使不同神经元之间的区分性更好。
（3）固定值初始化。

基于固定方差的参数初始化

为了降低 固定方差 对网络性能以及优化效率的影响，基于固定方差的随机初始化方法一般需要配合逐层归一化来使用。

基于方差缩放的参数初始化

根据神经元的连接数量来自适应地调整初始化分布的方差，这类方法称为方差缩放。

正交初始化

**7.5 逐层初始化

将传统机器学习中的数据归一化方法应用到深度神经网络中，对隐藏层的输入进行归一化，从而使得网络更容易训练。

原因：
更好的尺度不变性
更平滑的优化地形

常用的逐层归一化方法：批量归一化、层归一化、权重归一化和局部响应归一化。

批量归一化：可以对神经网络中任意的中间层进行归一化操作。

层归一化：对一个中间层的所有神经元进行归一化。可以应用在循环神经网络中。

权重归一化：对神经网络的连接权重进行归一化，通过再参数化方法，将连接权重分解为长度和方向两种参数。由于在神经网络中权重经常是共享的，权重数量往往比神经元数量要少，因此权重归一化的开销会比较小。

局部相应归一化：通常用在基于卷积的图像处理上。局部响应归一化和层归一化都是对同层的神经元进行归一化。不同的是，局部响应归一化应用在激活函数之后，只是对邻近的神经元进行局部归一化，并且不减去均值。和生物神经元的侧抑制现象比较类似。

7.6 超参数优化

常见的**超参数**有以下三类：
（1） 网络结构，包括神经元之间的连接关系、层数、每层的神经元数量、激活函数的类型等。
（2） 优化参数，包括优化方法、学习率、小批量的样本数量等。
（3） 正则化系数。
超参数优化主要存在两方面的**困难**
（1）超参数优化是一个组合优化问题，无法像一般参数那样通过梯度下降方法来优化，也没有一种通用有效的优化方法
（2）评估一组超参数配置的时间代价非常高
对于超参数的配置，比较简单的方法有网格搜索、随机搜索、贝叶斯优化、动态资源分配和神经架构搜索。

**7.7 网络正则化（一节看完）

正则化是一类通过限制模型复杂度，从而避免过拟合，提高泛化能力的方法，比如引入约束、增加先验、提前停止等。

L1 和 L2 正则化，通过约束参数的 L1 和 L2 范数来减小模型在训练数据集上的过拟合现象。

权重衰减：在每次参数更新时，引入一个衰减系数。

提前停止：在使用梯度下降法进行优化时，我们可以使用一个和训练集独立的样本集合，称为验证集，并用验证集上的错误来代替期望错误。当验证集上的错误率不再下降，就停止迭代。

丢弃法：当训练一个深度神经网络时，可以随机丢弃一部分神经元来避免过拟合，每次丢弃的神经元是随机的。

数据增强：在数据量有限的情况下，可以通过数据增强来增加数据量。常见的方法有旋转、翻转、缩放、平移、加噪声。

标签平滑：在输出标签中添加噪声来避免模型过拟合。

第八章 注意力机制与外部记忆

通过借鉴人脑解决信息过载的机制，从两方面来提高神经网络处理信息的能力。一方面是注意力，通过自上而下的信息选择机制来过滤掉大量的无关信息，一方面是引入额外的外部记忆，优化神经网络的记忆结构来提高神经网络存储信息的容量。

注意力一般分为两种：

（1）自上而下的有意识的注意力，称为聚焦式注意力。

（2）自下而上的无意识的注意力，称为基于显著性的注意力。

8.2 注意力机制

作为一种资源分配方案，将有限的计算资源用来处理更重要的信息。有注意力分布和加权平均概念。

为了增强网络容量，我们可以引入辅助记忆单元，将一些和任务相关的信息**保存**在辅助记忆中，在需要时再进行读取，这样可以有效地增加网络容量．这个引入的辅助记忆单元一般称为**外部记忆**。以区别于循环神经网络的**内部记忆**（即**隐状态**）。

8.5.1 端到端记忆网络

采用一种可微的网络结构，可以多次从外部记忆中读取信息。在端到端记忆网络中，外部记忆单元是只读的。其中有多跳操作，让主网络和外部记忆进行多轮交互。

8.5.2 神经图灵机

主要由两个部件构成：控制器和外部记忆。外部记忆定义为矩阵，控制器为一个前馈或循环神经网络，神经图灵机里的外部记忆是可读的。有读操作和写操作，写操作有增加和删除。

第九章 无监督学习

指从无标签的数据中学习出一些有用的模式。如果监督学习是建立输入-输出之间的映射关系，那么无监督学习就是发现隐藏的数据中的有价值信息，包括有效的特征、类别、结构以及概率分布等。

典型的无监督学习问题可以分为以下几类：无监督特征学习、概率密度估计、聚类。

9.1 无监督特征学习

主成分分析（PCA）：是一种最常用的数据降维方法，使得在转换后的空间中数据的方差最大。可以作为监督学习的数据预处理方法，用来去除噪声并减少特征之间的相关性，但是它并不能保证投影后数据的类别可分性更好。提高两类可分性的方法一般为监督学习方法，比如线性判别分析（LDA）。

稀疏编码：每个神经元仅对处于其感受野中特定的刺激信号做出响应。

训练方法：给定一组 N 个输入向量，需要同时学习基向量 A 以及每个输入样本对应的稀疏编码。

稀疏编码训练过程一般用交替优化的方法进行。

1）固定基向量 A 对每个输入 X 计算其对应的最优编码。

2）固定上一步得到的编码，计算其最优的基向量。

稀疏编码的优点：

计算量 ：稀疏性带来的最大好处就是可以极大地降低计算量。

可解释性：因为稀疏编码只有少数的非零元素，相当于将一个输入样本表示为 少数几个相关的特征。这样我们可以更好地描述其特征，并易于理 解。

特征选择：稀疏性带来的另外一个好处是可以实现特征的自动选择，只选择和 输入样本相关的最少特征，从而可以更好地表示输入样本，降低噪 声并减轻过拟合

自编码器：通过无监督的方式来学习一组数据的有效编码。自编码器通过将一组数据映射到特征空间得到每个样本的编码，希望这组编码可以重构出原来的样本，自编码器结构可分为编码器与解码器两部分。

稀疏自编码器：自编码器除了可以学习低维编码外，也能学习高维的稀疏编码。假设中间隐藏层的维度大于输入样本的维度，并让中间隐藏层尽量稀疏，这就是稀疏自编码器。

堆叠自编码器：采用逐层训练来学习网络参数。

降噪自编码器：通过引入噪声来增加编码鲁棒性的自编码器，并提高模型的泛化能力。

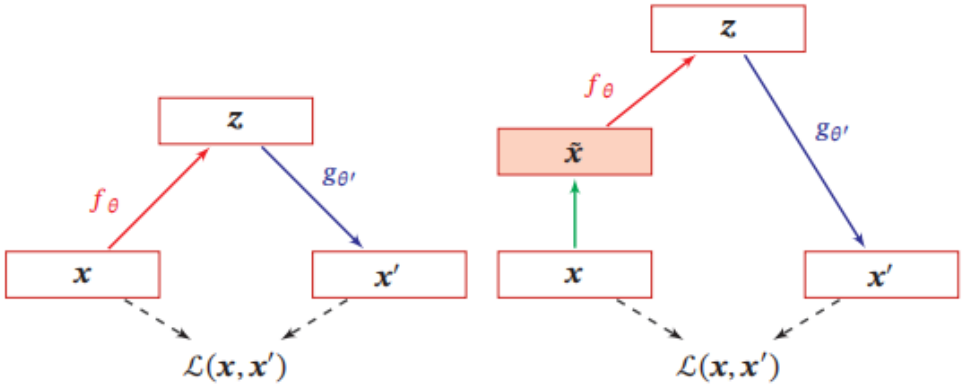


图 9.3　自编码器和降噪自编码器

概率密度估计

概率密度估计

- 参数密度估计（Parametric Density Estimation）
  - 根据先验知识假设随机变量服从某种分布，然后通过训练样本来估计分布的参数.
  - 估计方法：最大似然估计

$$\log p(\mathcal{D};\theta) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)};\theta).$$

- 非参数密度估计（Nonparametric Density Estimation）
  - 不假设数据服从某种分布，通过将样本空间划分为不同的区域并估计每个区域的概率来近似数据的概率密度函数。

参数密度估计一般存在以下问题

- 模型选择问题
  - 如何选择数据分布的密度函数?
  - 实际数据的分布往往是非常复杂的，而不是简单的正态分布或多项分布。
- 不可观测变量问题
  - 即我们用来训练的样本只包含部分的可观测变量，还有一些非常关键的变量是无法观测的，这导致我们很难准确估计数据的真实分布。
- 维度灾难问题
  - 高维数据的参数估计十分困难
  - 随着维度的增加，估计参数所需要的样本数量指数增加。在样本不足时会出现过拟合。

非参数密度估计

- 对于高维空间中的一个随机向量$\boldsymbol{x}$，假设其服从一个未知分布$p(\boldsymbol{x})$，则$\boldsymbol{x}$落入空间中的小区域$\mathcal{R}$的概率为

$$P = \int_{\mathcal{R}} p(\boldsymbol{x})d\boldsymbol{x}.$$

- 给定N个训练样本$D = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$，落入区域R的样本数量K服从二项分布

$$P_K = \binom{N}{K} P^K (1-P)^{1-K},$$

- 当N非常大时，我们可以近似认为

$$P \approx \frac{K}{N}$$

- 假设区域R足够小，其内部的概率密度是相同的，则有

$$P \approx p(\boldsymbol{x})V$$

- 结合上述两个公式，得到

$$p(\boldsymbol{x}) \approx \frac{K}{NV}$$

非参数密度估计

- 非参数密度估计(直方图方法除外)需要保留整个训练集。
- 而参数密度估计不需要保留整个训练集，因此在存储和计算上更加高效。

无监督学习

无监督学习

- 无监督学习是一种十分重要的机器学习方法．广义上讲，监督学习也可以看作一类特殊的无监督学习，即估计条件概率$p(y|\boldsymbol{x})$．条件概率$p(y|\boldsymbol{x})$可以通过贝叶斯公式转为估计概率$p(y)$和$p(\boldsymbol{x}|y)$，并通过无监督密度估计来求解；
- 无监督学习问题主要可以分为聚类、特征学习、密度估计等几种类型。
- 无监督特征学习是一种十分重要的表示学习方法．当一个监督学习任务的数据比较少时，可以通过大规模的无标注数据，学习到一种有效的数据表示，并有效提高监督学习的性能。
- 目前，无监督学习并没有像监督学习那样取得广泛的成功，其主要原因在于无监督学习缺少有效的客观评价方法，导致很难衡量一个无监督学习方法的好坏．无监督学习的好坏通常需要代入到下游任务中进行验证

第十章 模型独立的学习方式

需要一个模型可以快速地适应新的任务，因为每个任务都需要准备大量的训练数据。比如集成学习、协同学习、自训练、多任务学习、迁移学习、终身学习、小样本学习、元学习等．这里“模型独立”是指这些学习方式不限于具体的模型，不管是前馈神经网络、循环神经网络还是其他模型。

集成学习

▸三个臭皮匠赛过诸葛亮

定理 10.1：对于 M 个不同的模型 $f_1(\mathbf{x}), \cdots, f_M(\mathbf{x})$，平均期望错误为 $\bar{\mathcal{R}}(f)$。基于简单投票机制的集成模型 $f^{(c)}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x})$，其期望错误在 $\frac{1}{M}\bar{\mathcal{R}}(f)$ 和 $\bar{\mathcal{R}}(f)$ 之间。

为了增加模型之间的差异性，可以采取Bagging和Boosting这两类方法.

10.2 自训练和协同训练

利用少量标注数据和大量无标注数据进行学习的方式称为半监督学习：自训练和协同训练。

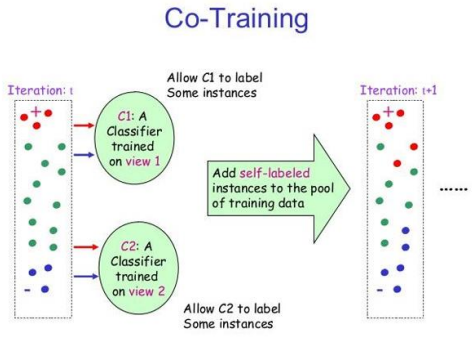
自训练是首先使用标注数据来训练一个模型，并使用这个模型来预测无标注样本的标签，把预测置信度比较高的样本及其预测的伪标签加入训练集，然后重新训练新的模型，并不断重复这个过程。

协同训练是自训练的一种改进方法，通过两个基于不同视角的分类器来互相促进。

协同训练（Co-Training）

▸Multi-View Learning

协同训练（Co-Training）是自训练的一种改进方法，通过两个基于不同视角（view）的分类器来互相促进



由于不同视角的条件独立性，在不同视角上训练出来的模型就相当于从不同视角来理解问题，具有一定的互补性．协同训练就是利用这种互补性来进行自训练的一种方法．首先在训练集上根据不同视角分别训练两个模型 $f1$ 和 $f2$，然后用 $f1$ 和 $f2$ 在无标注数据集上进行预测，各选取预测置信度比较高的样本加入训练集，重新训练两个不同视角的模型，并不断重复这个过程．

10.3 多任务学习

指同时学习多个相关任务，让这些任务在学习过程中共享知识，利用多个任务之间的相关性来改进模型在每个任务上的性能和泛化能力。

这四种常见的共享模式有：

（1）硬共享模式

（2）软共享模式

（3）层次共享模式

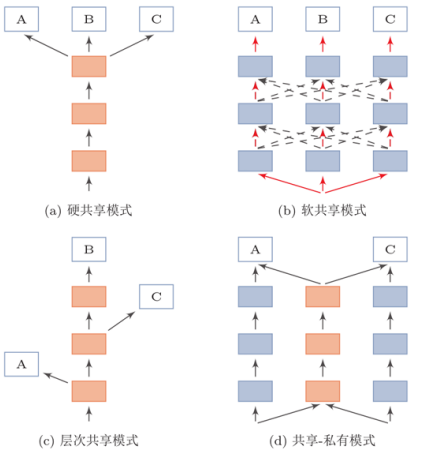
（4）共享—私有模式

多任务学习的流程可以分为两个阶段：

（1）联合训练阶段

（2）单任务精调阶段

多任务学习（Multitask Learning）



多任务学习（Multi-task Learning）是指同时学习多个相关任务，让这些任务在学习过程中共享知识，利用多个任务之间的相关性来改进模型在每个任务上的性能和泛化能力．多任务学习可以看作一种**归纳迁移学习**（Inductive Transfer Learning），即通过利用包含在相关任务中的信息作为归纳偏置（Inductive Bias）来提高泛化能力。

10.4 迁移学习

将相关任务的训练数据中的可泛化知识迁移到目标任务上。利用源领域中学到的知识来帮助目标领域上的学习任务。源领域的训练样本数量一般远大于目标领域

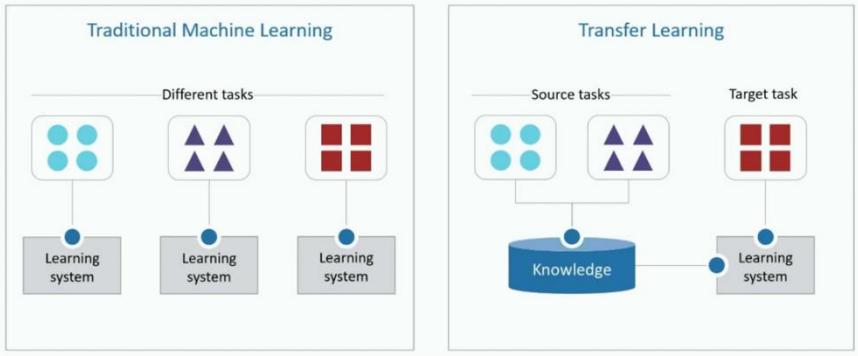
根据不同的迁移学习又分为归纳迁移学习和转导迁移学习。

归纳学习希望在训练数据集上学习到使得期望风险最小的模型；
转导学习的目标是学习一种在给定测试集上错误率最小的模型，在训练阶段可以利用测试集的信息。

归纳迁移学习是指在源领域和任务上学习出一般的规律，然后将这个规律迁移到目标领域和任务上；
而转导迁移学习是一种从样本到样本的迁移，直接利用源领域和目标领域的样本进行迁移学习。

迁移学习（Transfer Learning）

迁移学习是指两个不同领域的知识迁移过程，利用源领域（Source Domain）$\mathcal{DS}$ 中学到的知识来帮助目标领域（Target Domain）$\mathcal{DT}$ 上的学习任务．源领域的训练样本数量一般远大于目标领域．



迁移学习（Transfer Learning）

迁移学习根据不同的迁移方式又分为两个类型：

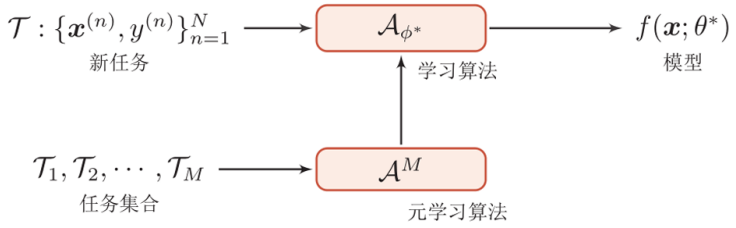
- **归纳迁移学习（Inductive Transfer Learning）**：一般的机器学习都是指归纳学习，即希望在训练数据集上学习到使得期望风险（即真实数据分布上的错误率）最小的模型．归纳迁移学习是指在源领域和任务上学习出一般的规律，然后将这个规律迁移到目标领域和任务上；
- **转导迁移学习（Transductive Transfer Learning）**：转导学习的目标是学习一种在给定测试集上错误率最小的模型，在训练阶段可以利用测试集的信息．而转导迁移学习是一种从样本到样本的迁移，直接利 用源领域和目标领域的样本进行迁移学习．

在**归纳迁移学习**中，由于源领域的训练数据规模非常大，这些预训练模型通常有比较好的泛化性，其学习到的表示通常也适用于目标任务．归纳迁移学习一般有下面**两种迁移方式**：

- **基于特征的方式：**将预训练模型的输出或者是中间隐藏层的输出作为特征直接加入到目标任务的学习模型中．目标任务的学习模型可以是一般的浅层分类器（比如支持向量机等）或一个新的神经网络模型；
- **精调的方式：**在目标任务上复用预训练模型的部分组件，并对其参数进行精调（Fine-Tuning）．

元学习（Meta Learning）

根据没有免费午餐定理，没有一种通用的学习算法可以在所有任务上都有效．因此，当使用机器学习算法实现某个任务时，我们通常需要“就事论事”，根据任务的特点来选择合适的模型、损失函数、优化算法以及超参数．那么，我们是否可以有一套自动方法，根据不同任务来动态地选择合适的模型或动态地调整超参数呢？事实上，人脑中的学习机制就具备这种能力．在面对不同的任务时，人脑的学习机制并不相同．即使面对一个新的任务，人们往往也可以很快找到其学习方式．这种可以动态调整学习方式的能力，称为**元学习（Meta-Learning），也称为学习的学习**（Learning to Learn）[Thrun et al., 2012]．



终身学习

虽然深度学习在很多任务上取得了成功，但是其前提是训练数据和测试数据的分布要相同，一旦训练结束模型就保持固定，不再进行迭代更新．并且，要一个模型同时在很多不同任务上都取得成功依然是一件十分困难的事情．由于不断的知识累积，人脑在学习新的任务时一般不需要太多的标注数据．

终身学习（Lifelong Learning），也叫持续学习（Continuous Learning），是指像人类一样具有持续不断的学习能力，根据历史任务中学到的经验和知识来帮助学习不断出现的新任务，并且这些经验和知识是持续累积的，不会因为新的任务而忘记旧的知识