人工智能技术
Artificial Intelligence

——人工智能:经典智能+智能计算+学习智能
AI: Classical Intelligence + Computing Intelligence + Learning Intelligenc

王鸿鹏，王润花，韩明静，
许丽，靖智博
南开大学人工智能学院
hanmj@mail.nankai.edu.cn

深度学习
DEEP LEARNING

深度学习通过梯度下降算法学习多层迭代的网络架构，它在人工智能的主要子领域中具有重要影响力

研究背景

深度学习（deep learning）是机器学习中一系列技术的组合，它的假设具有复杂代数电路的形式，且其中的连接强度是可调整的。“深度”的含义是指电路通常被设计成多层（layer），这意味着从输入到输出的计算路径包含较多计算步骤。

起源：用计算电路模拟大脑神经元网络的工作

神经网络：通过深度学习方法训练的网络

优势：

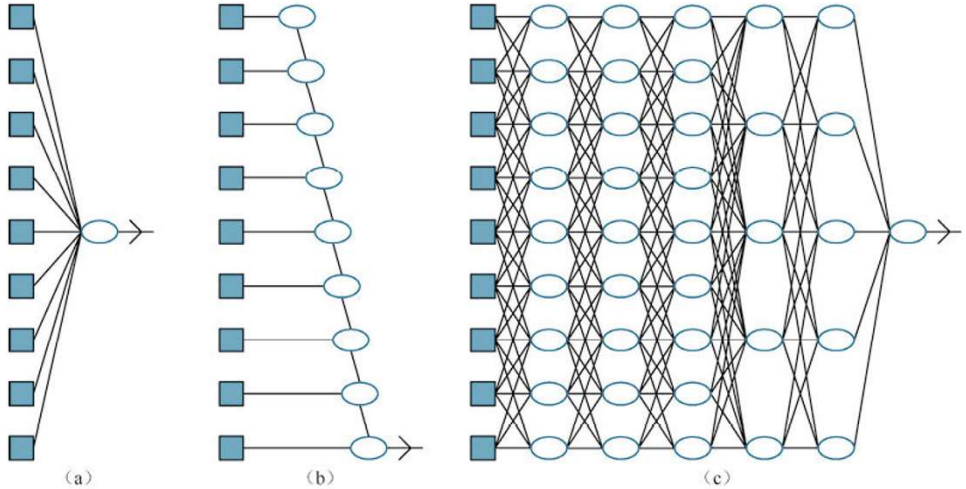
- 处理图像等高维数据时具有显著优势

——输入到输出的计算路径长

- 不同的输入变量既影响输出又相互影响

——增强了模型的表达能力

深度学习的基本思想是训练电路，使其计算路径可以很长，进而使得所有输入变量之间以复杂方式相互作用。

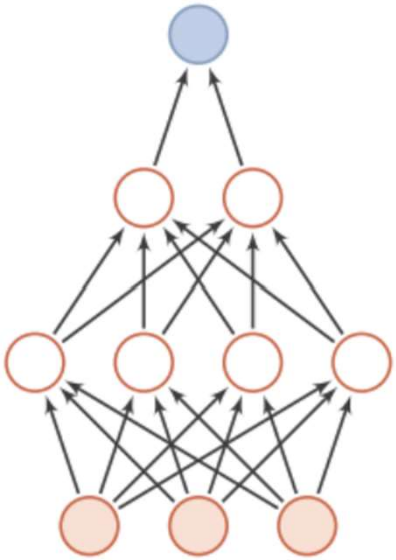


（a）浅层模型，例如线性回归，其输入到输出之间的计算路径很短。
（b）决策列表网络（19.5节）中可能存在某些具有长计算路径的输入，但大多数计算路径都较短。
（c）深度学习网络具有更长的计算路径，且每个变量都能与所有其他变量相互作用

简单前馈网络

前馈神经网络（feedforward network）

- 只在一个方向上有连接的网络
- 是一个有向无环图且有指定的输入和输出节点
- 每个节点计算一个关于输入的函数，并将结果传递给网络中的后续节点。信息从输入节点流向输出节点从而通过网络，且没有环路
- 输入层、隐藏层、输出层



例：布尔电路
输入：0或1
节点：关于输入的简单布尔函数
输出：0或1

任务：

神经网络中节点的一部分输入也可能是网络的参数，网络通过调整这些参数值，使网络整体拟合训练数据，以此来进行学习。

简单前馈网络

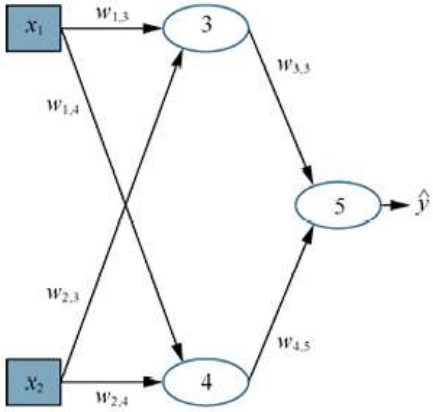
网络作为复杂函数

- 网络中的每个节点称为一个单元
- 一个单元将计算来自前驱节点的输入的加权和，并使用一个非线性的函数产生该节点的输出
- 令a_j为单元j的输出，并令w_{ij}为从单元i到单元j的连接的权重，有

$$a_j = g_j\left(\sum_i w_{i,j} a_i\right) \equiv g_j(in_j),$$

- g_j为用于单元j的非线性激活函数，in_j是单元j的输入的加权和
- 规定每个单元都有一个来自虚拟单元0的额外输入，这个来自虚拟单元0的输入固定为+1，并且该输入有权重w_{0j}，则可表示成向量形式

$$a_j = g_j(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x})$$



- 激活函数是非线性的这一事实非常重要，因为如果它不是非线性的，那么任意多个单元组成的网络将仍然只能表示一个线性函数。这种非线性使得由足够多的单元组成的网络能够表示任意函数。
- 万能近似（universal approximation）定理表明，一个网络只要有两层计算单元，且其中第一层是非线性的，第二层是线性的，那么它就可以以任意精度逼近任何连续函数。

简单前馈网络

不同的损失函数

- 逻辑斯谛函数或sigmoid函数

$$\sigma(x) = 1/(1 + e^{-x})$$

- ReLU函数，ReLU是修正线性单元（rectified linear unit）的简写

$$\mathrm{ReLU}(x) = \max(0, x)$$

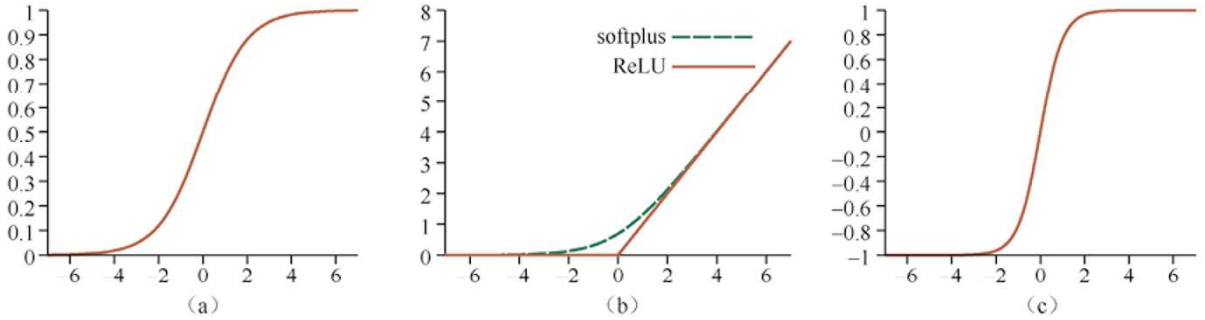
- softplus函数，它是ReLU函数的光滑版本：
  - softplus函数的导数为sigmoid函数.

$$\mathrm{softplus}(x) = \log(1 + e^{x})$$

- tanh函数：

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

可以发现tanh函数的值域为(−1, +1)，tanh函数是sigmoid经过伸缩与平移后的版本，即h(*x*) = 2*σ*(2*x*) − 1.

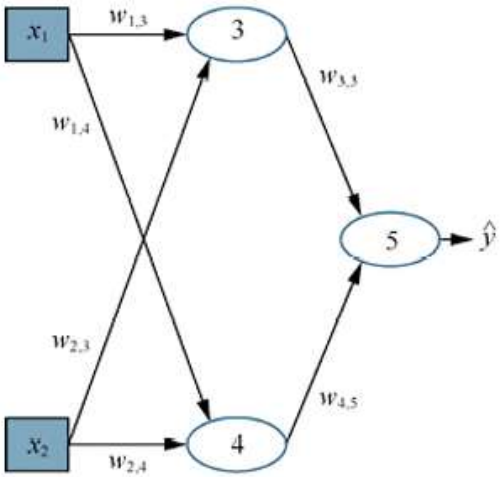


深度学习系统中常用的激活函数：
（a）逻辑斯谛函数或sigmoid函数。
（b）ReLU函数和softplus函数。
（c）tanh函数

简单前馈网络

把网络看作复杂函数：代数表达式组合

将多个单元组合到一个网络中会产生一个复杂的函数，它是由单个单元表示的代数表达式的组合：

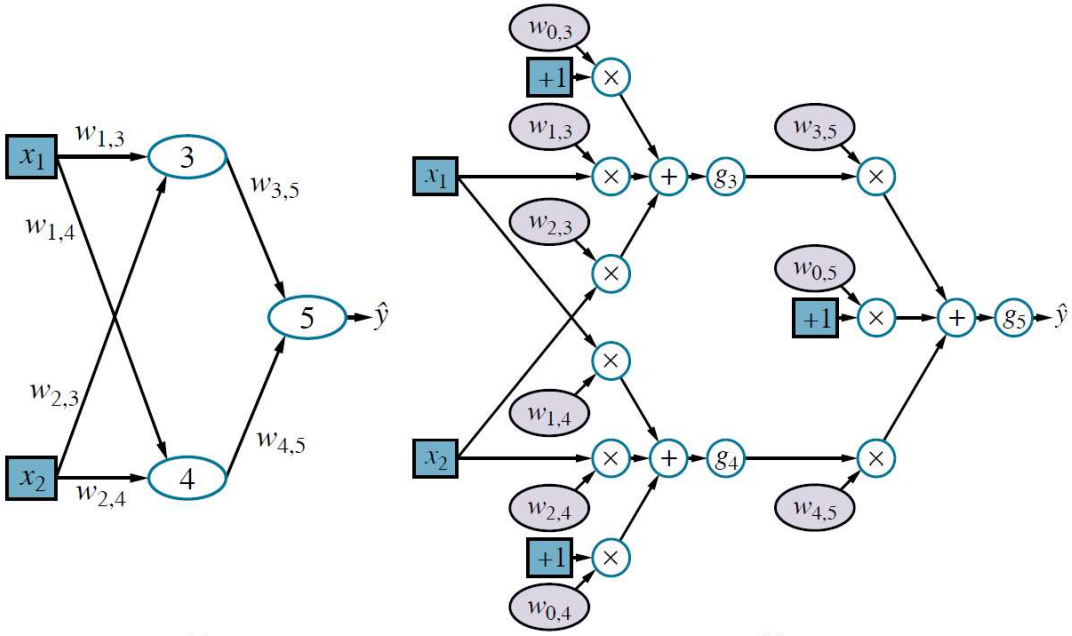


$$\begin{aligned}\hat{y} &= g_5(in_5) = g_5(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4) \\ &= g_5(w_{0,5} + w_{3,5}g_3(in_3) + w_{4,5}g_4(in_4)) \\ &= g_5(w_{0,5} + w_{3,5}g_3(w_{0,3} + w_{1,3}x_1 + w_{2,3}x_2) \\ &\qquad + w_{4,5}g_4(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2))\end{aligned}$$

输出表示为关于输入和权重的函数h_w(x)

简单前馈网络

把网络看作一个计算图或数据流图



- 通过构造计算图并调整其权重以拟合数据
- 全连接图：每一层中的每个节点都与下一层中的每个节点存在连接

（a）具有两个输入、一个包含两个单元的隐藏层和一个输出单元的神经网络，其中虚拟输入及其权重没有在图中给出。
（b）将（a）中的网络分解为完整的计算图

简单前馈网络

梯度与学习

输出层：根据输出计算其单元对应的权重的梯度
隐藏层：与输出没有直接联系，其单元对应的权重的梯度计算复杂

- 网络输出的预测为 $\hat{y} = h_w(x)$，真实值为y，使用平方损失函数L2，那么有

$$Loss(h_w) = L_2(y, h_w(x)) = \|y - h_w(x)\|^2 = (y - \hat{y})^2$$

链式法则：$\partial g(f(x))/\partial x = g'(f(x))\,\partial f(x)/\partial x$

- 连接到输出单元的权重

$$\begin{aligned}
\frac{\partial}{\partial w_{3,5}} Loss(h_w) &= \frac{\partial}{\partial w_{3,5}}(y-\hat{y})^2 = -2(y-\hat{y})\frac{\partial \hat{y}}{\partial w_{3,5}} \\
&= -2(y-\hat{y})\frac{\partial}{\partial w_{3,5}} g_5(in_5) = -2(y-\hat{y})g_5'(in_5)\frac{\partial}{\partial w_{3,5}} in_5 \\
&= -2(y-\hat{y})g_5'(in_5)\frac{\partial}{\partial w_{3,5}}(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4) \\
&= -2(y-\hat{y})g_5'(in_5)a_3
\end{aligned}$$

令 $\Delta_5 = 2(\hat{y}-y)\,g_5'(in_5)$　$\Delta_3 = \Delta_5 w_{3,5} g_3'(in_3)$

➡ 损失函数关于$w_{3,5}$的梯度为 $\Delta_5 a_3$，关于$w_{1,3}$的梯度为 $\Delta_3 x_1$

- 与输出单元没有直接联系的权重

$$\begin{aligned}
\frac{\partial}{\partial w_{1,3}} Loss(h_w) &= -2(y-\hat{y})g_5'(in_5)\frac{\partial}{\partial w_{1,3}}(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4) \\
&= -2(y-\hat{y})g_5'(in_5)w_{3,5}\frac{\partial}{\partial w_{1,3}} a_3 \\
&= -2(y-\hat{y})g_5'(in_5)w_{3,5}\frac{\partial}{\partial w_{1,3}} g_3(in_3) \\
&= -2(y-\hat{y})g_5'(in_5)w_{3,5}g_3'(in_3)\frac{\partial}{\partial w_{1,3}} in_3 \\
&= -2(y-\hat{y})g_5'(in_5)w_{3,5}g_3'(in_3)\frac{\partial}{\partial w_{1,3}}(w_{0,3} + w_{1,3}x_1 + w_{2,3}x_2) \\
&= -2(y-\hat{y})g_5'(in_5)w_{3,5}g_3'(in_3)x_1
\end{aligned}$$

简单前馈网络

梯度与学习

令 $\Delta_5 = 2(\hat{y} - y)\, g_5'(in_5)$，

则，损失函数关于$w_{3,5}$的梯度为$\Delta_5 a_3$

- 如果Δ_5是正的，这意味着$\hat{y}$过大（g’ 总是非负的）；如果 a_3也是正的，那么增大 $w_{3,5}$只会让结果变得更糟，而如果a_3是负的，那么增大 $w_{3,5}$会减少 误差。a_3的大小也很重要：如果在这个训练样例中 a_3很小，那么 $w_{3,5}$在产 生误差方面并不是主要的，也不需要做太大改变

令 $\Delta_3 = \Delta_5 w_{3,5} g_3'(in_3)$

则，损失函数关于$w_{1,3}$的梯度为$\Delta_3 a_1$

- 因此，单元3关于输入的感知误差为单元5关于输入的感知误差乘以从单元5返回到单元3的路径的信息。

反向传播：输出的误差通过网络进行回传的方式

$g_j'(in_j)$：局部导数. 如果导数很小或为0，这意味着修改与单位 j 相关的权重对其输出的影响可以忽略不计。层数较多的深度网络可能会遭遇 梯度消失（vanishing gradient）——误差信号通过网络进行反向传播时完全消失。

简单前馈网络

输入编码

- 计算图的输入和输出节点是指与输入数据x和输出数据y直接连接的节点
- 因子化数据
- 如果属性是布尔值，那么将有n个输入节点；通常，false映射为输入0，true映射为输入1，（有时也会使用-1 和 +1）
- 对于数值属性，无论是它是整数值还是实值，通常都按原样使用；如果不同样例之间的数量级存在较大差别，那么可以将这些值映射到对数尺度下。
- 对于具有两个以上取值范围的类别属性，通常采用所谓的独热编码（one-hot encoding）对其进行编码。
  - 具有d个可能值的属性由d个独立的输入位表示
  - 对任意给定的值，相应的输入位被设置为1，剩下的其他位将被设置为0
- 图像数据不符合因子化数据的范畴

简单前馈网络

输出层与损失函数

数据的标签y：原始数据值

预测值$\hat{y}$：概率

损失函数：

- 平方差损失
- 负对数似然损失

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} - \sum_{j=1}^{N} \log P_{\boldsymbol{w}}(\boldsymbol{y}_j \mid \boldsymbol{x}_j)$$

- 最小化交叉熵损失：
  - 交叉熵（Cross-entropy）：$H(P, Q)$，是两个分布P和Q之间差异性的一种度量。

$$H(P,Q) = \mathbf{E}_{\mathbf{z}\sim P(\mathbf{z})}[\log Q(\mathbf{z})] = \int P(\mathbf{z}) \log Q(\mathbf{z}) d\mathbf{z}.$$

  - softmax 层
  - 对于给定的输入向量 in = $\langle in_1, \ldots, in_d \rangle$ 将输出一个 d维向量 。其中输出向量的第k个元素由下式给出：

$$\text{softmax}(\mathbf{in})_k = \frac{e^{in_k}}{\sum_{k'=1}^{d} e^{in_{k'}}}.$$

softmax函数输出一个元素和为1的非负向量

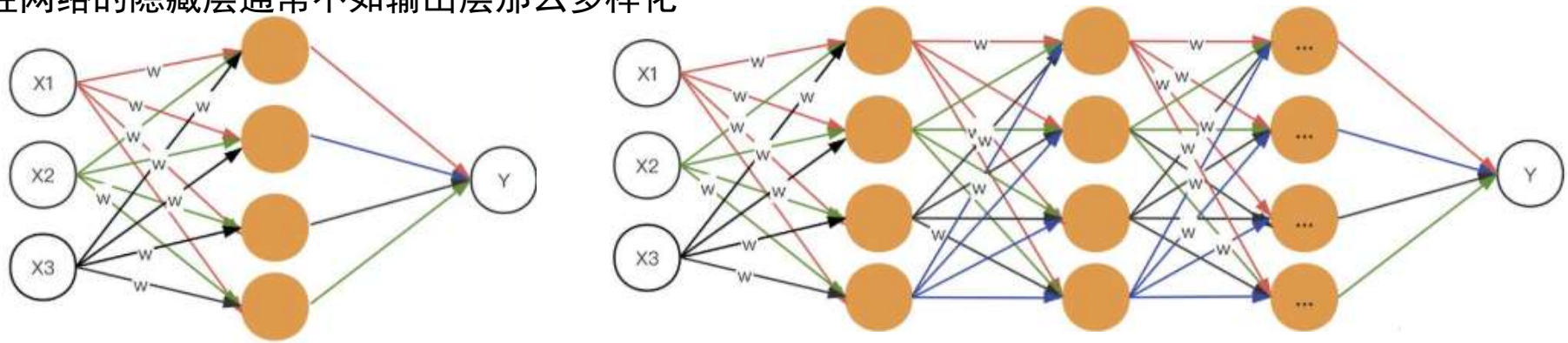
其他输出层：混合密度 （mixture density）层表示使用混合高斯分布的输出层

- 将预测每个混合成分的相对频率、 每个成分的平均值和每个成分的方差。只要这些输出值被损失函数恰 当地解释为真实输出值 y 的概率，那么在训练之后，网络将在由前一层给出的特征空间中拟合混合高斯模型

简单前馈网络

隐藏层

- 在处理输入向量x时，神经网络将执行若干次中间计算，最终输出 y
- 可以把网络每一层计算得到的值看作输入x的不同表示
- 每一层都将把前一层生成的表示转换为新的表示
- 在形成这些内部转换的过程中，深层网络经常可以发现一些有意义的数据的中间表示
- 神经网络的隐藏层通常不如输出层那么多样化



卷积网络

图像数据处理

- 不能把图像简单看作输入为像素值的向量——像素之间的邻接关系非常重要
- 庞大的参数空间需要大量的训练图像和庞大的计算预算来运行训练算法

设计思路：每个隐藏单元只接收来自图像的一个小局部区域的输入

- 在局部上注重了邻接关系（如果后续层也具有相同的局部性，那么网络将在全局意义下注重邻接性）
- 减少了权重的数量（如果每个局部区域有$l \ll b$个像素，那么权重的总数量将为$ln \ll n^2$。）

- 希望网络在处理图像数据时，能在小尺度到中等尺度上表现出近似空间不变性 （spatial invariance）

设计思路：局部空间不变性可以通过将隐藏单元与一个局部区域的连接的l个权重限制为对于每个隐藏单元都相等来实现

- 对于隐藏单元i和j，权重$w_{1,i}, \ldots, w_{l,i}$,与权重$w_{1,j}, \ldots, w_{l,j}$相等）
- 隐藏单元能成为特征检测器，且无论特征出现在图像中哪个位置都能被检测到
- 对于图像的每个局部区域，我们可能有d个具有不同权重集合的隐藏单元。这意味着权重的总数将有dl个，这个数字不仅远小于n^2，而且实际上与图像大小n无关

卷积网络

卷积神经网络

Convolutional neural network,是一种至少在较浅的层中包含空间局部连接的网络，并且在每一层中，单元之间权重是相同的

核（Kernel）：多个局部区域内权重置为相同的权重模式
卷积（Convolution）：将核应用于图像像素（或后续层中的空间上有序的单元）的过程称为卷积

- 假设输入向量 x 的大小为n,它对应于一维图像中的n个像素，核向量k的大小为l.

- 用符号*表示卷积运算,运算定义为 $z = x * k$.

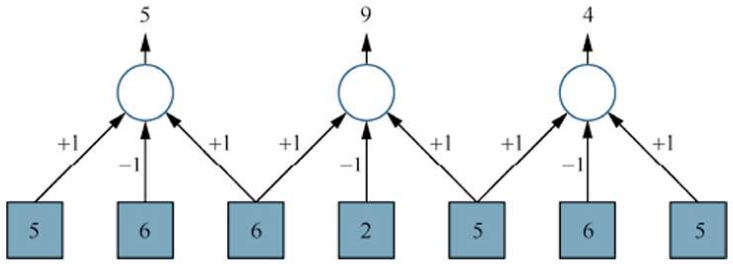
$$z_i = \sum_{j=1}^{l} k_j x_{j+i-(l+1)/2}.$$

→ 第i个位置的输出为：核k与x中以xi为中心、宽度为l的片段的内积

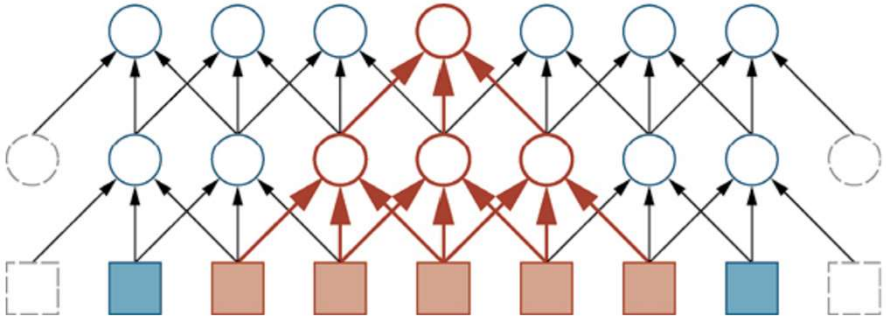
- 核的中心之间隔着的距离称核所用的步长（stride）s .

- 卷积过程在图像边缘停止，但是也可以用额外的像素对输入进行扩充（padding，可以是0，也可以等于外部像素），这样一来，核就可以精确地被应用n/s次。

卷积网络



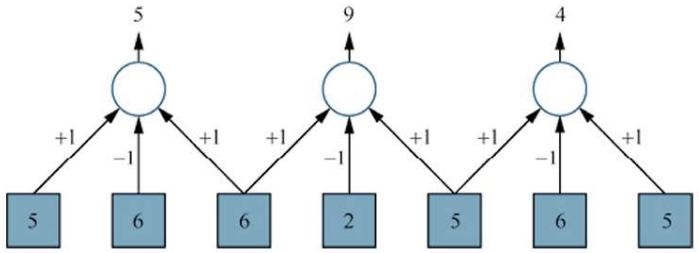
一维卷积运算的例子，其中核大小l = 3，步长s = 2。其中响应的峰值集中在较暗（光强较低）的输入像素上。在将结果输入下一个隐藏层之前，它们通常会经过一个非线性激活函数



一个处理一维图像数据的卷积神经网络的前两层，其核大小l = 3，步长s = 1。最左侧与最右侧作了填充，以使隐藏层与输入有相同的大小。红色所标记的区域是第二层隐藏层中某个单元的感受野。一般来说，越深的单元其感受野的范围越大

卷积网络

卷积操作可以看作矩阵乘法



$$\begin{pmatrix} +1 & -1 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & -1 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & -1 & +1 \end{pmatrix} \begin{pmatrix} 5 \\ 6 \\ 6 \\ 2 \\ 5 \\ 6 \\ 5 \end{pmatrix} = \begin{pmatrix} 5 \\ 9 \\ 4 \end{pmatrix}$$

- 在权重矩阵中，核出现在每一行，并按照步长相对于前一行进行移动，不必显式地构造权重矩阵（因为它大部分的位置为0）
- 卷积可以看作一个线性矩阵运算这一事实表明，梯度下降可以简单且高效地应用于CNN，就像它可以应用于普通的神经网络一样。
- CNN灵感最初来自于神经科学中提出的视觉皮层模型。在这些模型中，神经元的感受野（receptive field）是指感觉输入中能够影响神经元激活状态的部分。
- 在一个卷积神经网络中，第一个隐藏层中一个单元的感受野会很小——恰好为核的大小，即l个像素。
- 在网络的更深层中，一个单元的感受野会大得多。 当步长为1时，第m个隐藏层中的节点的感受野大小将为$(l-1)m$； 其增长速度关于m是线性的。当步长大于1时，第m 层中的每个像素将表示第 m−1层中的s个像素；因此，感受野将以$O(ls^m)$的速度增长，即与网络深度呈指数关系。

卷积网络

池化与下采样

神经网络中的池化（pooling）层用一个值来提取前一层中的一组相邻单元的信息。与卷积层类似，池化层也有一个大小为l，步长为s的核，但是它的运算方式是固定的，而不是学习得到的。通常来说，池化层不与激活函数相连接。

池化的两种常见的形式

- 平均池化：计算l个输入的平均值
  - 等价于采用一个均匀的核 $\boldsymbol{k} = [1/l, \cdots, 1/l]$ 进行卷积。如果令l=s，那么其效果将为粗化图像
  - 的分辨率——以s尺度进行下采样（downsample）
  - 在池化之后，一个占用10s个像素的对象将只占用10个像素
  - 平均池化有助于多尺度下的识别。
  - 降低了后续层中所需的权重数量，从而降低了计算成本并加快了学习速度
- 最大池化：计算l个输入的最大值.
  - 同样可以单纯用于下采样，语义上与平均池化有一些不同
  - 最大池化实现的是逻辑析取的运算，表明该单元的感受野内存在某个特征.

卷积网络

卷积神经网络的张量运算

- 在深度学习术语中，张量（tensor）可以是任意多维的数组。向量和矩阵是张量在一维和二维情况下的特例。
- 矩阵与向量的表示有助于保持数学推导的简洁与优雅，而且能提供关于计算图的简明描述
- 对于卷积神经网络，张量是一种跟踪数据在网络各层传输过程中的“形状”的表示方法
- 计算效率：如果将一个网络表述为张量运算的序列，那么深度学习软件包能生成底层计算结构高度优化的编译代码
- 深度学习的程序通常在GPU（图形处理器）或TPU（张量处理器）上运行，这使得高度并行运算成为可能

卷积网络

卷积神经网络的张量运算

已知：

输入张量大小：$H_{\text{in}} \times W_{\text{in}} \times C_{\text{in}}$
卷积核大小：$K_h \times K_w$
填充（padding）：P_h，P_w
步长（stride）：S_h，S_w
卷积核个数：C_{out}

卷积输出高宽公式：

$$H_{out} = \left\lfloor \frac{H_{in} - K_h + 2P_h}{S_h} \right\rfloor + 1$$

$$W_{out} = \left\lfloor \frac{W_{in} - K_w + 2P_w}{S_w} \right\rfloor + 1$$

输出通道数：$C_{out} = \text{filters}$

最终输出 $H_{out} \times W_{out} \times C_{out}$

示例：
假设将使用256像素×256像素的RGB图像进行训练，每个批量的大小为64。在这种情况下，输入将是一个四维张量，其大小为256×256×3×64。接着我们使用96个大小为5×5×3的核对其进行处理，其中x方向和y方向上的步长均为2。这使得输出张量的大小为128×128×96×64。

- 通常称这样的张量为特征映射（feature map），因为它所给出的是核在整个图像上提取出的特征。
- 在该例中，它由96个通道组成，其中每个通道携带一个特征的信息。注意，和输入张量不同，该特征映射不再拥有专门的颜色通道；
- 如果学习算法发现颜色对于网络最终的预测有帮助，它仍可能出现在各个特征通道中

卷积网络

残差网络

残差网络（residual network）是用来构造深层网络且同时避免度消失问题的一种流行且成功的方法。

- 典型的深层模型网络的每一层将前一层的表示完全取代，所以每一层所做的操作必须有意义

$$z^{(i)} = f(z^{(i-1)}) = g^{(i)}(W^{(i)}z^{(i-1)})$$

$z(i)$：第i层的单元的值

- 残差网络的核心思想在于，它认为每一层应当对前一层的表示进行扰动，而不是完全替换它. 如果学习到的扰动较小，那么后一层的输出将接近于前一层的输出。

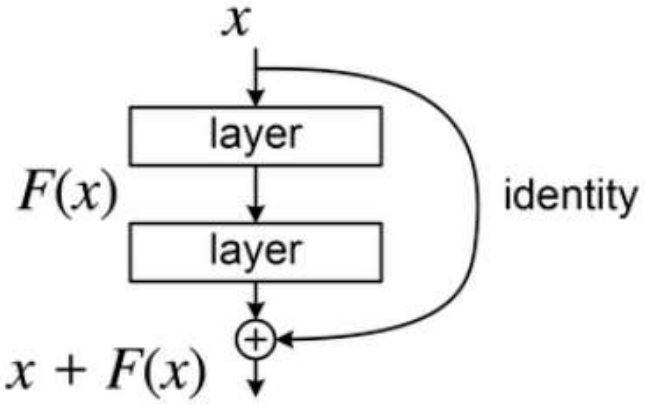
$$z^{(i)} = g_r^{(i)}(z^{(i-1)} + f(z^{(i-1)}))$$

g_r：残差层的激活函数

f看作残差，它对从第i-1层传递到第i层的默认信息进行扰动。通常选择带有一个非线性层与一个线性层的神经网络作为计算残差的函数：

$$f(z) = Vg(Wz)$$

其中W和V为学习到的带一般偏差权重的权重矩阵。



卷积网络

残差网络

残差网络使得有效地学习一个极度深层的网络成为可能：

考虑将某一层置为V= 0以使该层失效会发生什么：残差f将会消失

$$z^{(i)} = g_r(z^{(i-1)})$$

假设g_r由ReLU激活函数构成，z^{i-1}同样也为关于输入的ReLU函数

$$z^{(i-1)} = \mathrm{ReLU}(in^{(i-1)})$$

在这种情况下，有

$$z^{(i)} = g_r(z^{(i-1)}) = \mathrm{ReLU}(z^{(i-1)}) = \mathrm{ReLU}(\mathrm{ReLU}(in^{(i-1)})) = \mathrm{ReLU}(in^{(i-1)}) = z^{(i-1)}$$

在以ReLU为激活函数的残差网络中，一个权重为零的层将仅仅把它的输入原封不动地输出。网络的其他部分可以认为这一层不存在。 对于残差网络，传递信息是其固有的特征，然而传统的网络必须学习如何传递信息，并且可能会因为参数选取不当从而导致灾难性的失败

学习算法

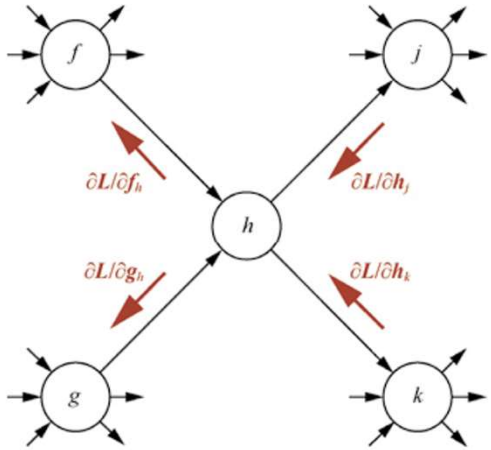
计算图中的梯度计算

- 梯度可以通过将误差信息从网络的输出层反向传播到隐藏层来计算
- 反向传播过程将消息沿着网络中的每个连接传回。在每个节点上，传入的消息将被收集起来，并在计算后将新消息传递给下一层。这些消息均为损失函数L的偏导数
- 例如，后向消息 $\partial L / \partial h_j$，为损失函数L关于j的第一个输入（即从h到j的前向消息）的偏导数。现在，节点h通过节点j和节点k对损失函数L产生影响，因此有

$$\partial L / \partial h = \partial L / \partial h_j + \partial L / \partial h_k$$

- 节点h可以收集来自节点j和节点k的传入消息并计算损失函数L关于h的导数。现在，为计算传出消息 $\partial L / \partial f_h$ 与 $\partial L / \partial g_h$，需要用到

$$\frac{\partial L}{\partial f_h} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial f_h} \text{ 和 } \frac{\partial L}{\partial g_h} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial g_h}$$

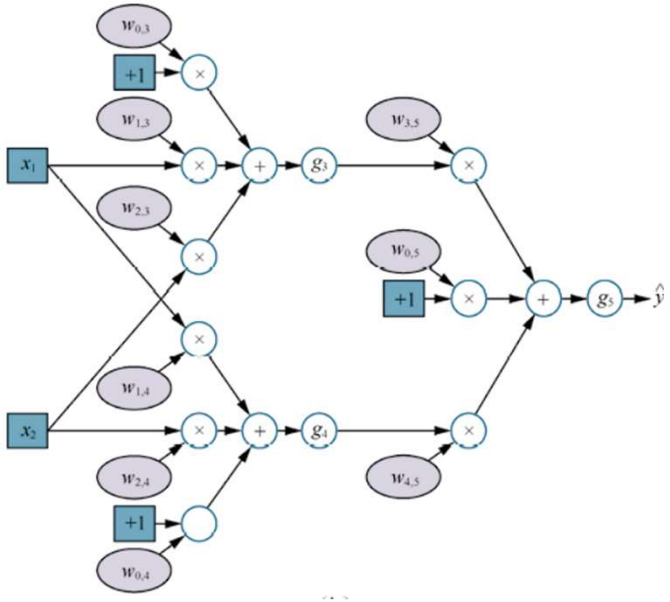


学习算法

计算图中的梯度计算

反向传播的过程从输出节点出发，其中每个初始消息 $\partial L / \partial \hat{y}_j$ 是根据预测值$\hat{y}$和训练数据中的真实值y直接从L的表达式计算得到的。在每个内部节点，传入的后向消息据式1进行求和，传出消息将根据式2得到。在该计算图中的每个代表权重w的节点处，该过程都将终止。
此时，关于w的传入消息的总和即为 $\partial L / \partial w$ ——恰好是我们更新w所需的梯度

- 权重共享通过将每个共享权重的节点视为计算图中具有多重输出的单个节点来实现. 在反向传播过程中，这将导致有多重的传入梯度消息。这意味着关于被共享的权重的梯度是网络中相关联的每个位置的梯度贡献的总和。
- 从对反向传播过程的描述中，可以清楚地看出，它的计算代价与计算图中的节点数呈线性关系，这与前向计算的代价是一样的。
- 反向传播的一个缺点是，它需要存储正向传播期间所计算的大部分中间值，以便计算反向传播中的梯度。这意味着训练网络的总内存开销与整个网络中的单元数成正比。因此，尽管网络本身可以仅用具有大量循环的代码隐式地表示，而不需要根据数据的结构显式地表示，但反向传播的代码仍要求显式地存储所有的中间结果。



学习算法

批量归一化

- 批量归一化（batch normalization）是一种常用的技巧，它通过对每个小批量样例在网络内部层生成的值进行重新缩放来提高SGD的收敛速度。

- 考虑网络中的某个节点z，节点z关于其中m个样例的输出值为$z_1,\ldots,z_m$。批量归一化将每一个z_i替换为一个新的值$\hat{z}_i$

$$\hat{z}_i = \gamma \frac{z_i - \mu}{\sqrt{\epsilon + \sigma^2}} + \beta,$$

其中 μ 是小批量中的z的均值， σ是 $z_1, \ldots, z_m$的标准差， ϵ是一个用于防止除法中分母为零的较小的常数，γ 和 β是可学习的参数.

- 批量归一化根据 β 和 γ值，对中间量的均值和方差进行归一化 .

如果没有批量归一化

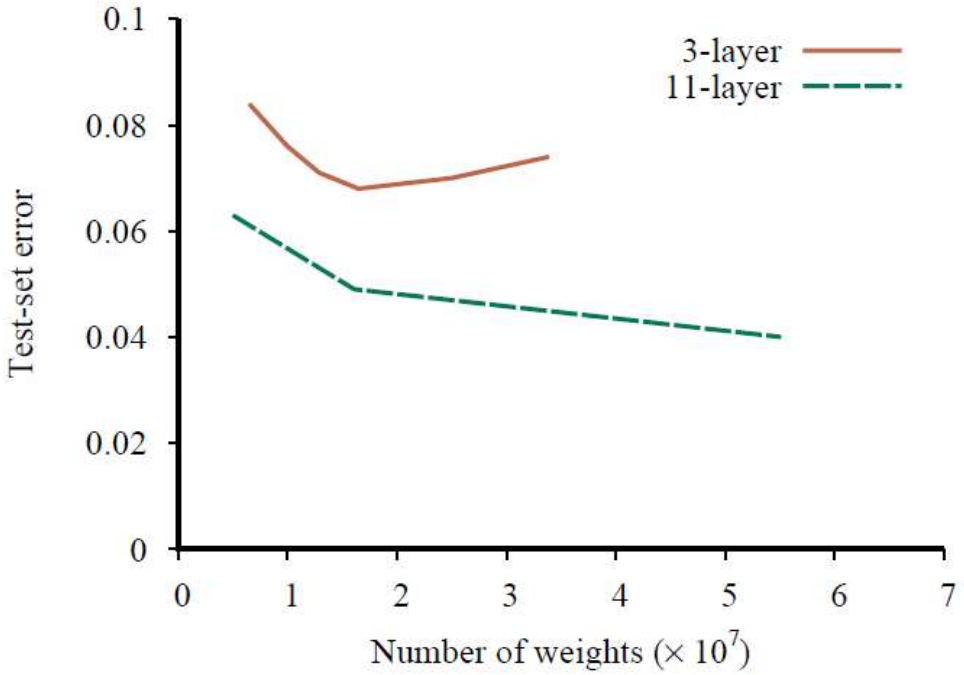
- i如果某一层的权重非常小，并且该层的标准差减小至接近0，该层的信息就会丢失

泛化

选择正确的网络架构

- 一些神经网络架构被显式设计成对特定类型数据具有较好的泛化性能（例如，预期卷积架构在图像问题上有更好的泛化性能，循环网络在处理文本和音频信号时有更好的泛化性能。）
- 当两个网络的权重数量接近时，更深的网络通常具有更好的泛化性能
- 深度学习系统对于具有高维输入的图像、视频、语音信号等任务，它们的性能优于任何其他纯粹的机器学习方法
- 深度学习模型缺乏在一阶逻辑和上下文无关语法中看到的组合与量化表达能力。
- 深度学习模型可能产生某些不直观的错误。它们倾向于产生不连续的输入-输出映射，因此对输入的一个小幅度扰动可能导致输出产生一个较大的扰动。

泛化



3层和11层卷积网络的测试集误差与层的宽度（权重的总数）的关系。图中使用的数据来源于早期版本的谷歌系统，该系统用于把街景车所拍摄的照片中的地址进行转录（Goodfellow et al., 2014）

泛化

神经架构搜索

- 神经架构搜索（neural architecturesearch）来探索可能的网络架构的状态空间
- 进化算法：对网络进行重组（将两个网络的一部分连接在一起），也可以进行变异（添加或移除一层或更改一个参数值）。
- 爬山算法也可以与变异操作一起使用
- 一个主要的困难是估计候选网络的价值
  - 在多批量的测试集上对其进行训练，并在验证集上评估其准确性
  - 启发式评价函数

泛化

权重衰减

- weight decay
- 权重衰减类似于正则化
- 权重衰减的方式包括对用于训练神经网络的损失函数添加惩罚项

$$\lambda \sum_{i,j} W_{i,j}^2$$

λ 是控制惩罚强度的超参数，求和是对网络中的所有权重进行的

- 权重衰减实现了一种最大后验（MAP）学习

$$\begin{aligned} h_{\mathrm{MAP}} &= \operatorname*{argmax}_{\mathbf{w}} P(\mathbf{y} \mid \mathbf{X}, \mathbf{W}) P(\mathbf{W}) \\ &= \operatorname*{argmin}_{\mathbf{w}} [-\log P(\mathbf{y} \mid \mathbf{X}, \mathbf{W}) - \log P(\mathbf{W})] . \end{aligned}$$

其中第一项是一般的交叉熵损失；第二项倾向于选择先验分布下可能性较高的权重。如果令

$$\log P(\boldsymbol{W}) = -\lambda \sum_{i,j} W_{i,j}^2$$

这意味着P(W)是一个零均值高斯先验

泛化

暂退法

通过干预网络以减少测试集误差的方法是暂退法（dropout），在训练的每一步中，暂退法都会随机选择单元的子集并令其停用，从而创建一个新网络，并在新网络中应用一步反向传播学习其代价是使得网络更难拟合训练集。

- 假设所使用的是批量大小为 m 的随机梯度下降。对于每个批量，暂退算法将以下过程应用于网络的每个节点：每个单元的输出以概率p乘以因子1/ p；否则，该单元的输出将固定为零。暂退法通常应用于隐藏层，且采用 p = 0.5；对于输入单元，通常 p = 0.8效果最好。这个过程将产生一个单元数量接近原网络一半的简化网络，并应用批量大小为m的训练样例在该网络上进行反向传播。该过程将以这样的方式不断进行，直到训练完成。在测试阶段，模型将不采用暂退法。
- 从以下几个角度看待暂退法
  - 通过在训练时引入噪声，这将迫使模型对噪声具有健壮性。
  - 暂退法近似了精简网络的大规模集成
  - 通过暂退法训练得到的隐藏单元必须学会成为有用的隐藏单元；它们还必须学会与众多其他可能的隐藏单元集合相兼容，这些隐藏单元集合可能包含在完整模型中，也可能不包含在完整模型中。
  - 暂退法应用于深度网络靠后的层中，这迫使模型做出的最终决策更加健壮，这是因为网络将更加关注样例所有的抽象特征，而不是只关注其中一个而忽略其他特征。
- 暂退法迫使模型为每一个输入学习多个健壮的解释

循环神经网络

循环神经网络（recurrent neural network，RNN）

- 允许计算图中存在环

每个环都将有一个延迟，使得单元可以把在较早的时间步中所得的输出作为输入

（在没有延迟的情况下，循环电路可能会达到不一致的状态。）

- RNN具有内部状态或记忆（memory）：在较早时间步中接收的输入会影响RNN对当前输入的响应
- 与前馈网络相比，RNN有了更强的表达能力:网络的隐藏状态 $z_t = f_w(z_{t-1}, x_t)$ 捕获所有先前输入的信息

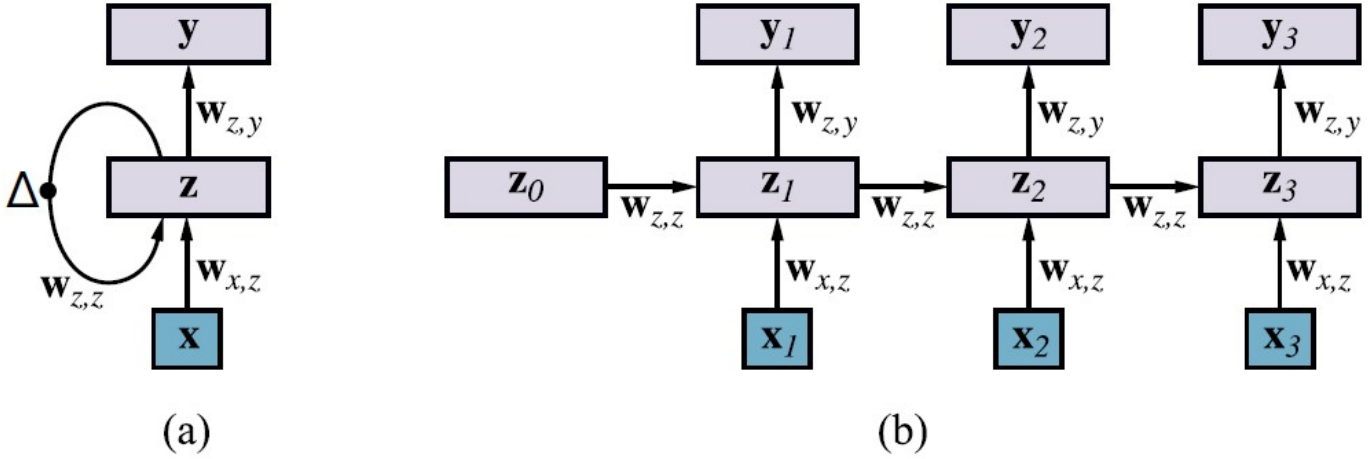
训练基本的循环神经网络

- 考虑一个基本的RNN模型，它有一个输入层x、一个具有循环连接的隐藏层z和一个输出层y。假设在每个时间步中，我们都观测到训练数据x和y。以下等式定义的模型给出了时间步t的变量值

$$\begin{aligned}\mathbf{z}_t &= f_{\mathbf{W}}(\mathbf{z}_{t-1}, \mathbf{x}_t) = \mathbf{g}_z(\mathbf{W}_{z,z}\mathbf{z}_{t-1} + \mathbf{W}_{x,z}\mathbf{x}_t) \equiv \mathbf{g}_z(\mathbf{in}_{z,t}) \\ \hat{\mathbf{y}}_t &= \mathbf{g}_y(\mathbf{W}_{z,y}\mathbf{z}_t) \equiv \mathbf{g}_y(\mathbf{in}_{y,t}),\end{aligned}$$

$\mathbf{g}_z$，$\mathbf{g}_y$分别表示隐藏层和输出层的激活函数.

循环神经网络



（a）基本的RNN模型的示意图，其中隐藏层z具有循环连接，符号Δ表示延迟。
（b）同一网络在3个时间步上展开以创建前馈网络。注意，权重在所有时间步中是共享的

循环神经网络

循环神经网络

为了使计算式简洁，给出一个只有一个输入单元、一个隐藏单元和一个输出单元的RNN的梯度计算。在这种情况下，偏差项为

$$z_t = g_z(w_{z,z}z_{t-1} + w_{x,z}x_t + w_{0,z}) \text{与} \hat{y}_t = g_y(w_{z,y}z_t + w_{0,y})$$

$$\begin{aligned}
\frac{\partial L}{\partial w_{z,z}} &= \frac{\partial}{\partial w_{z,z}}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 = \sum_{t=1}^{T}-2(y_t - \hat{y}_t)\frac{\partial \hat{y}_t}{\partial w_{z,z}} \\
&= \sum_{t=1}^{T}-2(y_t - \hat{y}_t)\frac{\partial}{\partial w_{z,z}}g_y(in_{y,t}) = \sum_{t=1}^{T}-2(y_t - \hat{y}_t)g_y'(in_{y,t})\frac{\partial}{\partial w_{z,z}}in_{y,t} \\
&= \sum_{t=1}^{T}-2(y_t - \hat{y}_t)g_y'(in_{y,t})\frac{\partial}{\partial w_{z,z}}(w_{z,y}z_t + w_{0,y}) \\
&= \sum_{t=1}^{T}-2(y_t - \hat{y}_t)g_y'(in_{y,t})w_{z,y}\frac{\partial z_t}{\partial w_{z,z}}
\end{aligned}$$

隐藏单元z_t的梯度为

$$\begin{aligned}
\frac{\partial z_t}{\partial w_{z,z}} &= \frac{\partial}{\partial w_{z,z}}g_z(in_{z,t}) = g_z'(in_{z,t})\frac{\partial}{\partial w_{z,z}}in_{z,t} = g_z'(in_{z,t})\frac{\partial}{\partial w_{z,z}}(w_{z,z}z_{t-1} + w_{x,z}x_t + w_{0,z}) \\
&= g_z'(in_{z,t})\left(z_{t-1} + w_{z,z}\frac{\partial z_{t-1}}{\partial w_{z,z}}\right)
\end{aligned}$$

循环神经网络

循环神经网络

可以观察到：

- 第一，梯度的表达式是循环的：计算时间步t对梯度的贡献要用到时间步t－1的贡献。如果我们以正确的方式对计算进行排序，那么梯度计算的总运行时间将与网络的大小呈线性关系。这种算法被称为基于时间的反向传播（back propagation through time，BPTT）
- 第二，如果我们对该循环计算进行迭代，可以注意到时间步T的梯度将包括与 $w_{z,z}\prod_{t=1}^{T} g'_z(in_{z,t})$ 成正比的项。
  - 对于sigmoid、tanh和ReLU激活函数，它们有导数g’ ≤1，因此，如果$w_{z,z}$ ＜1，简单RNN必然会面临梯度消失的问题；
  - 如果$w_{z,z}$ ＞1，可能会面临梯度爆炸（exploding gradient）的问题

循环神经网络

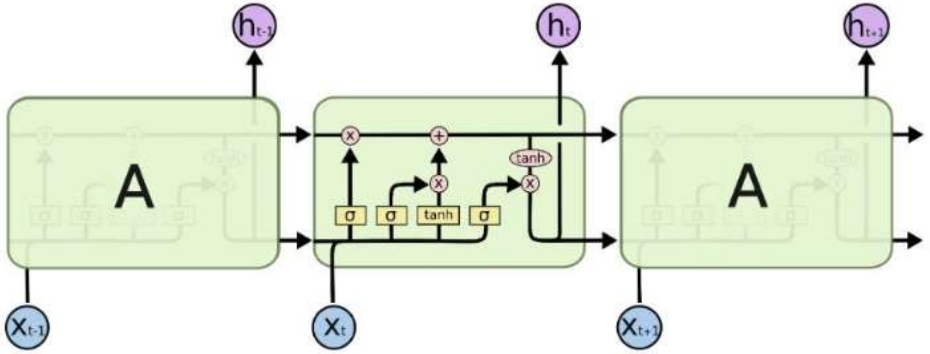
长短期记忆RNN

- LSTM的长期记忆成分，称为记忆单元（memory cell），用字母c表示，基本上从一个时间步被复制到另一个时间步。新信息以直接加入记忆的方式进行更新，这样一来，梯度表达式将不会随着时间的推移进行乘法累积
- 使用了门单元（gatingunit），它是控制LSTM中信息流的一个向量，通过控制相应信息向量的逐个元素相乘来实现

- 门的类型:
  - 遗忘门f决定了记忆单元中的每个元素是被记住了（将复制到下一个时间步）还是被遗忘了（重置为0）
  - 输入门i决定了记忆单元中的每个元素是否被来自当前时间步的输入向量的新信息进行加法
  - 输出门o决定了记忆单元中的每个元素是否被转移到短期记忆z，它在基本的RNN中起着与隐藏状态类似的作用

- LSTM中的门是宽泛的不局限于布尔函数，门单元的值始终在[0, 1]范围内，并且它是当前输入和先前隐藏状态经过sigmoid函数而获得的输出，更新公式为

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_{x,f}\mathbf{x}_t + \mathbf{W}_{z,f}\mathbf{z}_{t-1}) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_{x,i}\mathbf{x}_t + \mathbf{W}_{z,i}\mathbf{z}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{x,o}\mathbf{x}_t + \mathbf{W}_{z,o}\mathbf{z}_{t-1}) \\
\mathbf{c}_t &= \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i}_t \odot \tanh(\mathbf{W}_{x,c}\mathbf{x}_t + \mathbf{W}_{z,c}\mathbf{z}_{t-1}) \\
\mathbf{z}_t &= \tanh(\mathbf{c}_t) \odot \mathbf{o}_t ,
\end{aligned}
$$



无监督学习与迁移学习

无监督学习

无监督学习算法仅从无标签的输入x中进行学习

- 学习新的表示法
  （例如图像的新特征）
- 学习一个生成模型
  （从中生成新的样本）

目标：学习联合分布P(x, z)，其中z是一组隐变量，即以某种方式表达数据x内容的未观测变量。能同时实现表示学习（它从原始的x向量中构造出有意义的z向量）与生成模型如果对 P(x, z)中的z变量进行积分，将得到P(x)

无监督学习与迁移学习

无监督学习

概率主成分分析（ PPCA）：一个简单的生成模型

- probabilistic principal components analysis (PPCA)
- z取自一个零均值的球形高斯分布，x 通过z乘以权重矩阵W并添加球形高斯噪声来生成

$$P(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})$$
$$P_W(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x};\mathbf{W}\mathbf{z},\sigma^2\mathbf{I})$$

权重W（以及可选的噪声参数 σ^2）可以通过最大化数据的似然学习得到，具体为，

$$P_W(\mathbf{x}) = \int P_W(\mathbf{x},\mathbf{z})d\mathbf{z} = \mathcal{N}(\mathbf{x};\mathbf{0},\mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}).$$

关于W的最大化问题可以通过梯度方法或者高效的EM迭代算法进行求解。学习得到W后，新的数据样本可以由PW（x）直接生成。此外，若某个新观测数据x有非常低的概率，它可以被标记为潜在的异常数据。

- 通常假设z的维数远小于x的维数，这样模型就可以尽可能地用少量的特征来解释数据

无监督学习与迁移学习

自编码器

自编码器是一个由两部分组成的模型

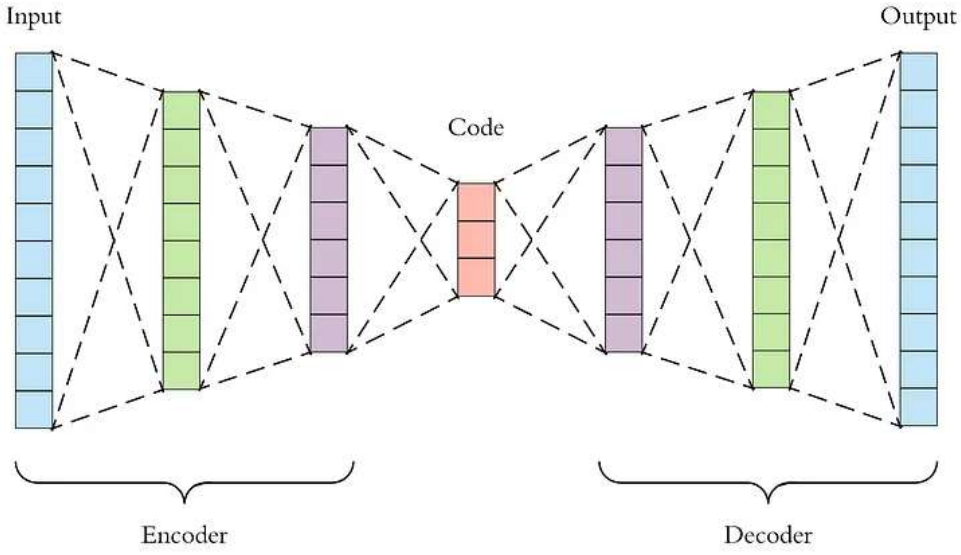
- 一个从x映射到表示 $\hat{\mathbf{z}}$ 编码器
- 一个从表示 $\hat{\mathbf{z}}$ 映射到观测数据x的解码器.

一般来说，编码器是一个参数化函数f，解码器是一个参数化函数g。通过对模型进行训练，使得x≈g(f(x))

线性编码器，其中f和g均为线性函数且共享一个权重矩阵 $\mathbf{W}$:

$$\hat{\mathbf{z}} = f(\mathbf{x}) = \mathbf{W}\mathbf{x}$$
$$\mathbf{x} = g(\hat{\mathbf{z}}) = \mathbf{W}^{\top}\hat{\mathbf{z}}.$$

- 主成分分析(PCA)模型是一个简单的生成模型，它对应于一个简单的线性自编码器



无监督学习与迁移学习

变分自编码器（variational autoencoder，VAE）

- 更复杂的自编码器来获得更复杂的生成模型
- 变分法:使用一个变分后验（variational posterior） $Q(z)$，作为真实后验分布的近似值，它来自某个在计算上易处理的分布族。
- KL散度: “尽可能”

$$D_{KL}(Q(\mathbf{z})\|P(\mathbf{z}|\mathbf{x})) = \int Q(\mathbf{z}) \log \frac{Q(\mathbf{z})}{P(\mathbf{z}|\mathbf{x})} d\mathbf{z},$$

- Q与P的对数比率（关于Q）的均值。 $D_{KL}(Q(z)\|P(z|x)) \geq 0$ 其中等号成立当且仅当Q与P完全相同

- 变分下界 $\mathcal{L}$（variational lower bound, or **ELBO**）

$$\mathcal{L}(\mathbf{x}, Q) = \log P(\mathbf{x}) - D_{KL}(Q(\mathbf{z})\|P(\mathbf{z}|\mathbf{x}))$$

$$\mathcal{L} = \log P(x) - \int Q(z) \log \frac{Q(z)}{P(z|x)} dz$$

$$= \int Q(z) \log Q(z) dz + \int Q(z) \log P(x) P(z|x) dz$$

$$= H(Q) + E_{z\sim Q} \log P(z, x)$$

变分自编码器提供了一种在深度学习场景中使用变分学习的方法。变分学习涉及最大化L关于P和Q的参数的过程。对于一个变分自编码器，解码器g(z)可以解释为 $\log P(x|z)$

无监督学习与迁移学习

深度自回归模型

自回归模型（AR model）:

- 自回归模型（autoregressive model，或AR model）意味着向量x的每个元素xi是基于向量其他元素进行预测得到的。
- 这种模型不含有隐变量。
- 如果x的大小是固定的，那么AR模型可以看作一个完全可观测且可能完全连通的贝叶斯网络
- 自回归模型最常见的一个应用是时序数据分析

深度自回归模型（Deep autoregressive model）:

- 将线性高斯模型替换为具有适当输出层的任意深度网络的模型，其具体形式取
- 决于x_t是离散的还是连续的。
- 用于语音生成的WaveNet模型

无监督学习与迁移学习

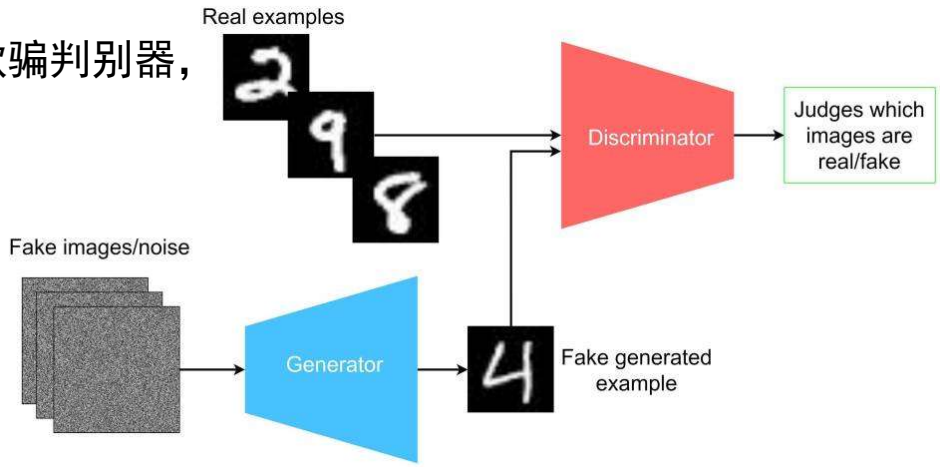
生成对抗网络络（generative adversarial network，GAN）

- 一对结合在一起形成生成系统的网络
  - 生成器（generator），将z值映射到x
  - 判别器（discriminator），一个经过训练的分类器，它用于判断输入的x为真（从训练集中获取的）或假（由生成器生成的）

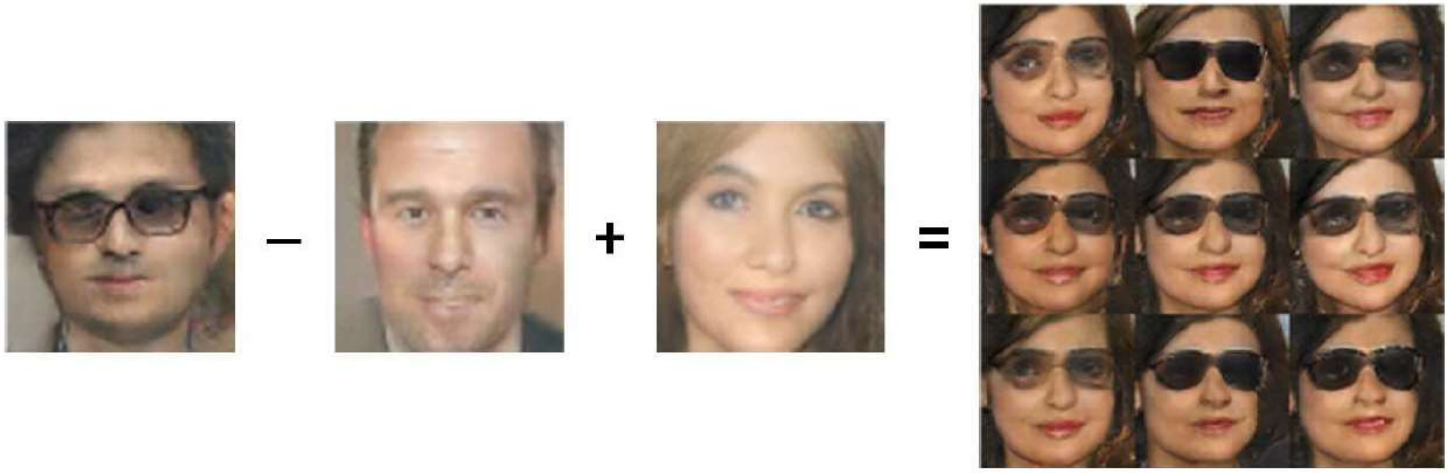
- 隐式模型（implicit model）：样本可以被生成，但其概率不易获得
- （在贝叶斯网络中，样本的概率是样本生成路径上条件概率的乘积）

- 生成器和判别器的训练是同时进行的：生成器将学习如何欺骗判别器，而判别器将学习如何准确区分真假数据。

- **应用：图像生成任务**



无监督学习与迁移学习



生成模型如何使用z空间中的不同方向来表示人脸不同方面的信息。实际上我们可以在z空间中进行运算。这里的图像都是从学习到的模型中生成的，并且图像解释了当我们解码z空间中的不同点时会发生什么。我们从“戴眼镜的男人”这个对象的坐标出发，减去“男人”的坐标，再加上“女人”的坐标，得到“戴眼镜的女人”的坐标。图像经许可摘自（Radford et al., 2015）

无监督学习与迁移学习

迁移学习和多任务学习

在迁移学习（transfer learning）中，一个学习任务的经验有助于智能体更好地学习另一个任务
对于神经网络，学习的过程即为调整权重，因此迁移学习最合理的方法是将任务A学习中所得到的权重复制到将用于任务B训练的网络。然后使用任务B的数据，通过一般的梯度下降等方式来更新权重。

选择任务时需要人类的专业知识介入
方式

- 冻结预训练模型的前几层：这些层起到了特征检测器的作用

- 新数据集将只修改较高层的参数：这些层用于识别特定问题的特征并对其进行分类

目前常用的做法是从一个预先训练好的模型，如 RoBERTa 模型，再对模型进行微调

微调方法:

- 举例说明所需领域中使用的专业
- 在模型将要完成的任务上对模型进行训练

多任务学习（multitask learning）是迁移学习的一种形式，在这种学习中，我们同时训练一个关于多个目标的模型。

应用

视觉

- AlexNet深度学习系统在2012年ImageNet竞赛中的大获成功
- ImageNet竞赛是一项监督学习任务，它共有120万幅图像，分为1000个不同的类别
- 自2012年以来，随着网络设计、训练方法和计算资源的改进，ImageNet竞赛的Top5错误率已降至2%以下——远低于一个受过培训的人的错误率（约5%）

自然语言处理

- 机器翻译
- 实现端到端学习的可能性、自动生成单词含义的内部表示以及学习到的编码器和解码器的可交换性
- 流水线方法的表现已经被由深度学习实现的端到端方法超越
- 网络功能的大部分进展都基于将单个单词重新表示为高维空间中的向量，即所谓的词嵌入

强化学习

- 决策型智能体从一系列奖励信号中进行学习，这些信号提供了其行为质量的一些信息
- 目标是优化未来奖励的总和
- 智能体可以学习一个价值函数、一个Q函数或者一个策略等。从深度学习的角度看来，所有这些内容都是
- 可以用计算图来表示的函数

小结

本章描述了学习由深度计算图表示的函数的方法。本章重点如下。

- 神经网络用参数化的线性阈值单元网络表示复杂的非线性函数。
- 反向传播算法实现了在参数空间中使用梯度下降以最小化损失函数。
- 深度学习适用于复杂环境中的视觉对象识别、语音识别、自然语言处理和强化学习。
- 卷积网络特别适用于图像处理和其他具有网格拓扑结构的数据处理任务。
- 循环网络对于包括语言建模和机器翻译在内的序列处理任务是有效的。