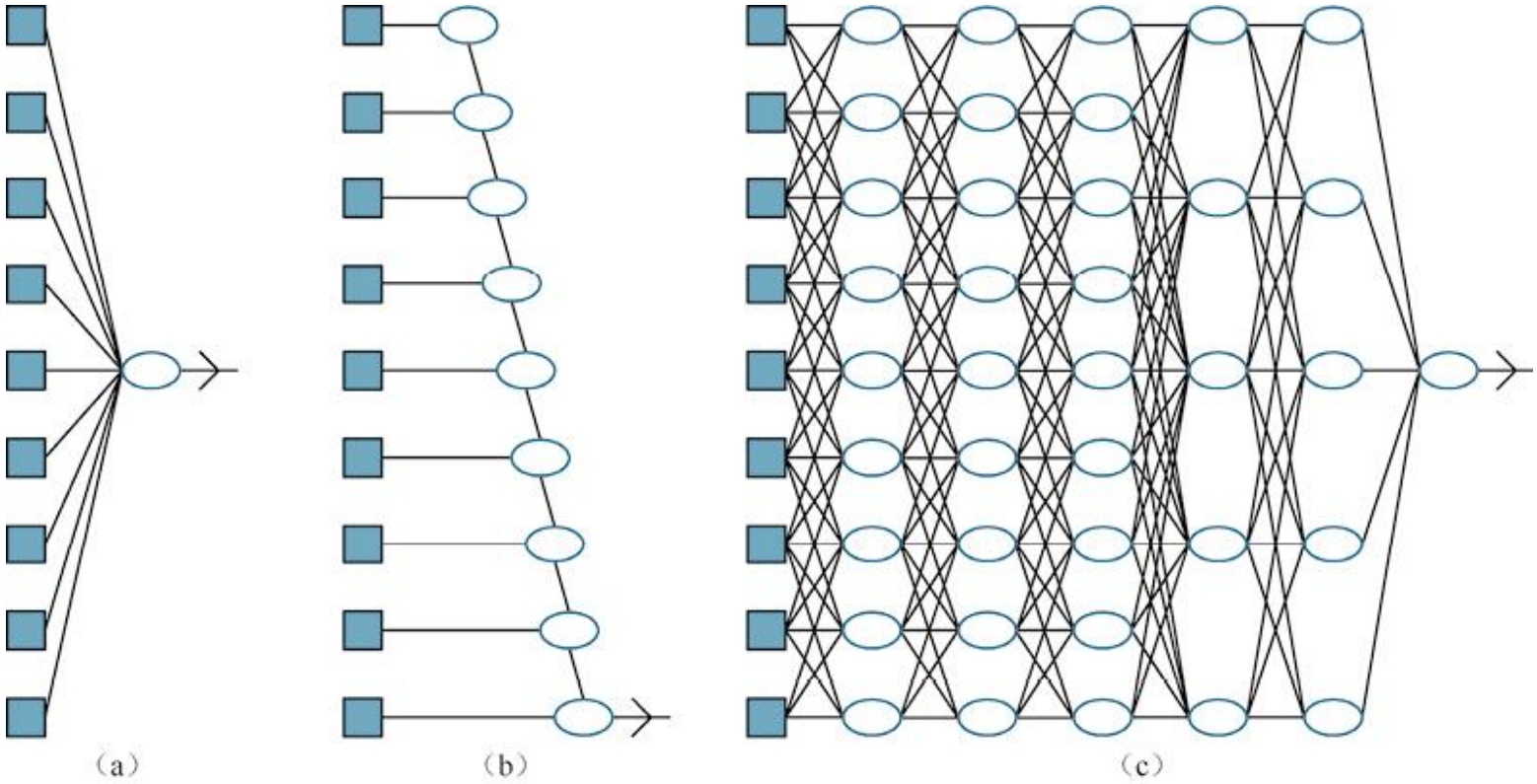
提纲

南京大学 NANJING UNIVERSITY

- **深度学习**
  - 简单前馈网络
  - 深度学习的计算图
  - 卷积网络
  - 学习算法
  - 泛化
  - 循环神经网络
  - 无监督学习与迁移学习

深度学习



（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

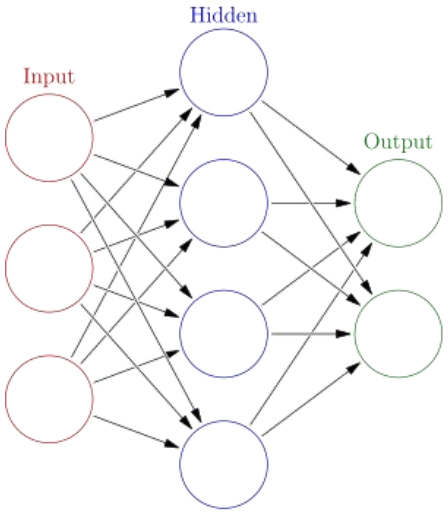
南京大学 NANJING UNIVERSITY

前馈网络

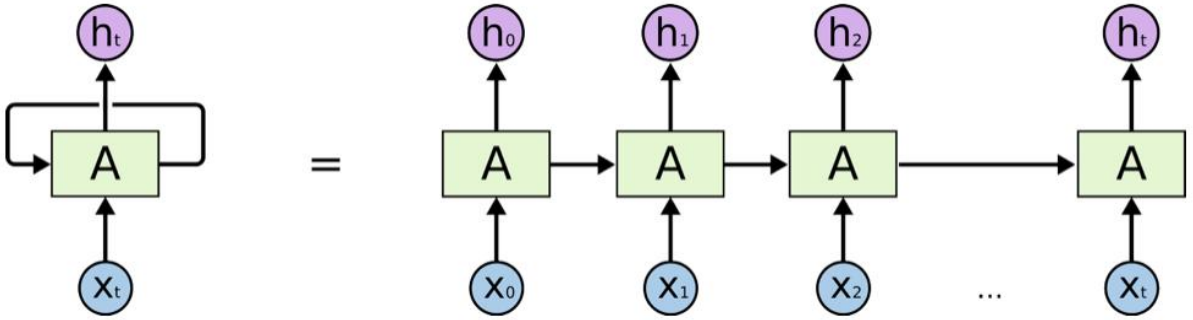
- 只在一个方向上有连接（从输入到输出）
- 有向无环图且有指定的输入和输出节点

循环网络

- 将其中间输出或最终输出反馈到自己的输入中
- 网络中的信号值将形成一个具有内部状态或记忆的动态系统



前馈网络



循环网络

简单前馈网络

南京大学 NANJING UNIVERSITY

网络作为复杂函数

- 网络中的每个节点称为一个单元（*unit*）
- 计算来自前驱节点的输入的加权和
- 使用一个非线性的函数产生该节点的输出

$$a_j = g_j\left(\sum_i w_{i,j} a_i\right) \equiv g_j(in_j) \quad \Rightarrow \quad a_j = g_j(\boldsymbol{w}^{\top}\boldsymbol{x})$$

- g_j 是一个非线性激活函数，a_j 为单元 j 的输出，$w_{i,j}$ 是从单元 i 到单元 j 的连接的权重。

- **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

南京大学 NANJING UNIVERSITY

网络作为复杂函数

不同的激活函数:

- **逻辑斯蒂**或*sigmoid*函数

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

- *ReLU*函数，修正线性单元（*Rectified Linear Unit*）的简写

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

- *softplus*函数，*ReLU* 函数的光滑版本:
  - *softplus* 函数的导数是 *sigmoid* 函数

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

- *tanh*函数:

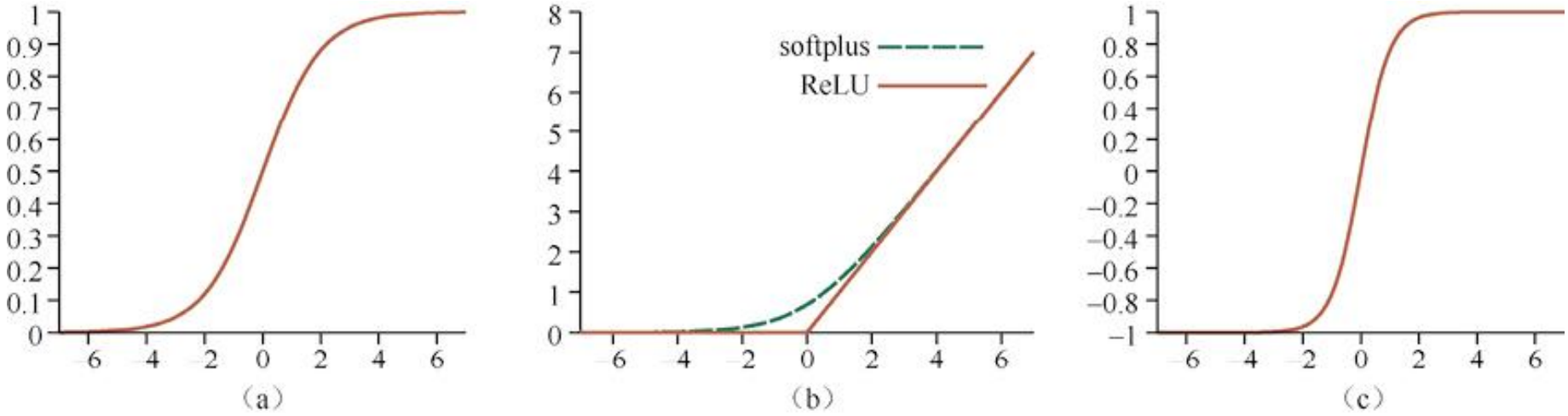
$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

tanh 函数的值域为 $(-1, +1)$，是*sigmoid*经过伸缩与平移后的版本,
即$\tanh(x) = 2\sigma(2x) - 1$.

简单前馈网络

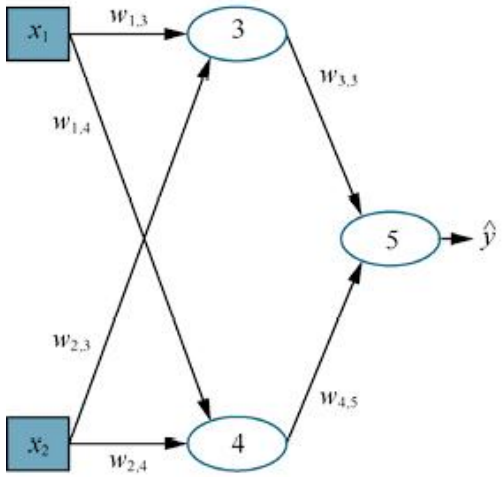
南京大学 NANJING UNIVERSITY

网络作为复杂函数



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

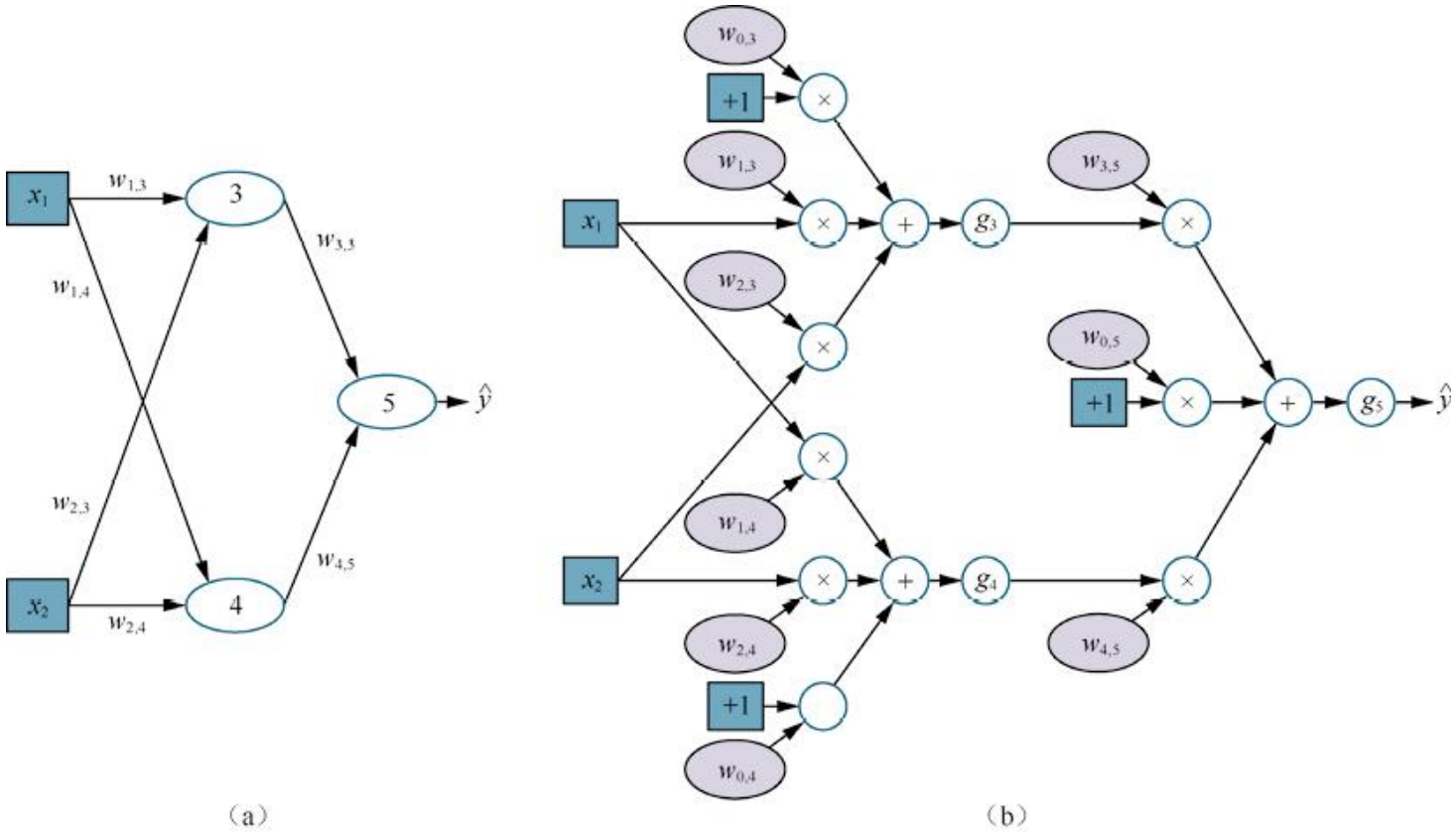


（a）

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

（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。

简单前馈网络



（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（computation graph）

简单前馈网络

南京大学 NANJING UNIVERSITY

梯度与学习

- 使用平方损失函数 L_2
- 网络输出的预测为 $\hat{y} = h_{\mathbf{w}}(\mathbf{x})$
- $Loss(h_{\mathbf{w}}) = L_2(y, h_{\mathbf{w}}(\mathbf{x})) = \|y - h_{\mathbf{w}}(\mathbf{x})\|^2 = (y - \hat{y})^2$

反向传播：输出的误差通过网络进行回传的方式

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

深度学习的计算图

南京大学 NANJING UNIVERSITY

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
- false映射为输入0，true映射为输入1，或者 (-1 和 +1)
- 对于数值属性，通常都按原样使用（可进行数值缩放）
- 对于具有两个以上取值范围的类别属性，通常采用独热（one-hot）编码
  - 具有d个可能值的属性由d个独立的输入位表示.
  - 相应的输入位被设置为1，剩下的其他位将被设置为0

深度学习的计算图

南京大学 NANJING UNIVERSITY

输出层与损失函数

大部分的深度学习应用：

- 常见的做法是将输出值 $\hat{\mathbf{y}}$ 表述为概率
- 使用**负对数似然**作为损失函数

最小化交叉熵损失函数.

- 交叉熵: $H(P, Q)$, 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
  - 对于一个给定的输入向量$\mathbf{in} = \langle in_1, \ldots, in_d \rangle$，输出一个$d$ 维向量

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

  - 输出一个元素和为1的非负向量

- 对于回归问题，即目标值y是连续值的问题，我们通常使用线性输出层

深度学习的计算图

南京大学 NANJING UNIVERSITY

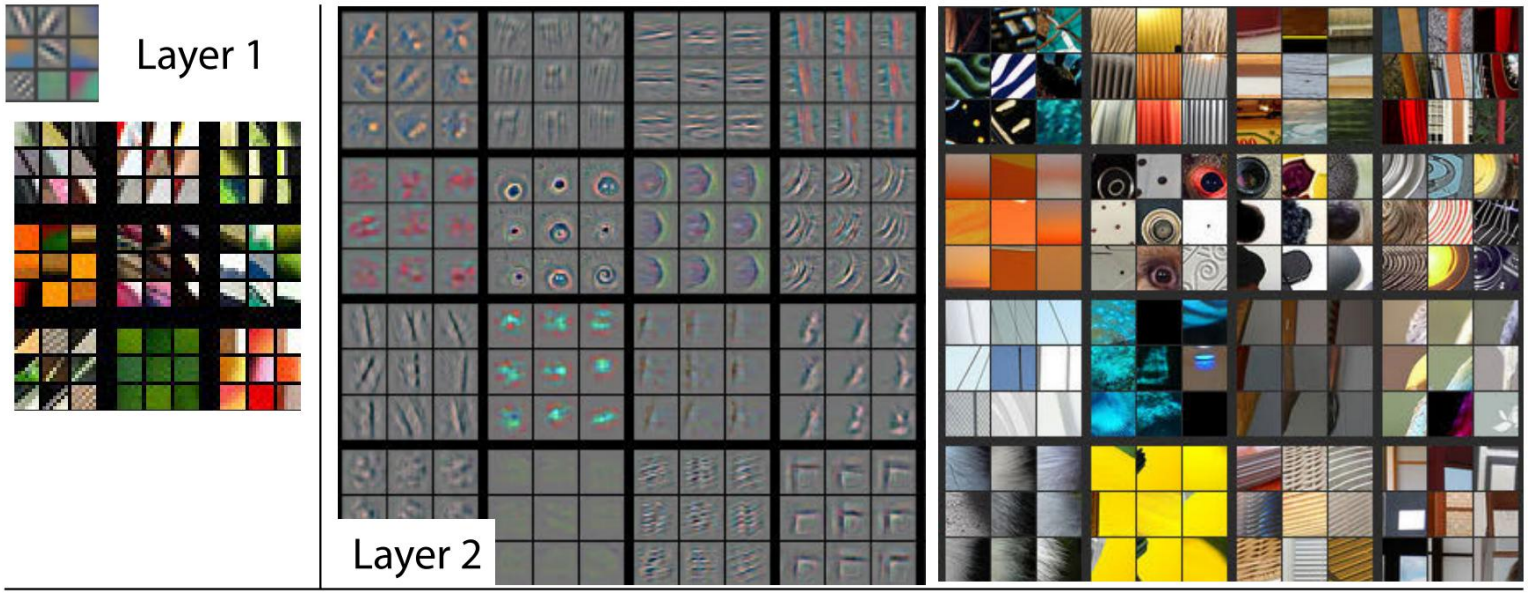
隐藏层

- 在产生输出*y*之前的中间计算。
- 网络每一层计算得到的值可以看作输入x的不同表示。
- 每一层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。

深度学习的计算图

南京大学 NANJING UNIVERSITY

隐藏层

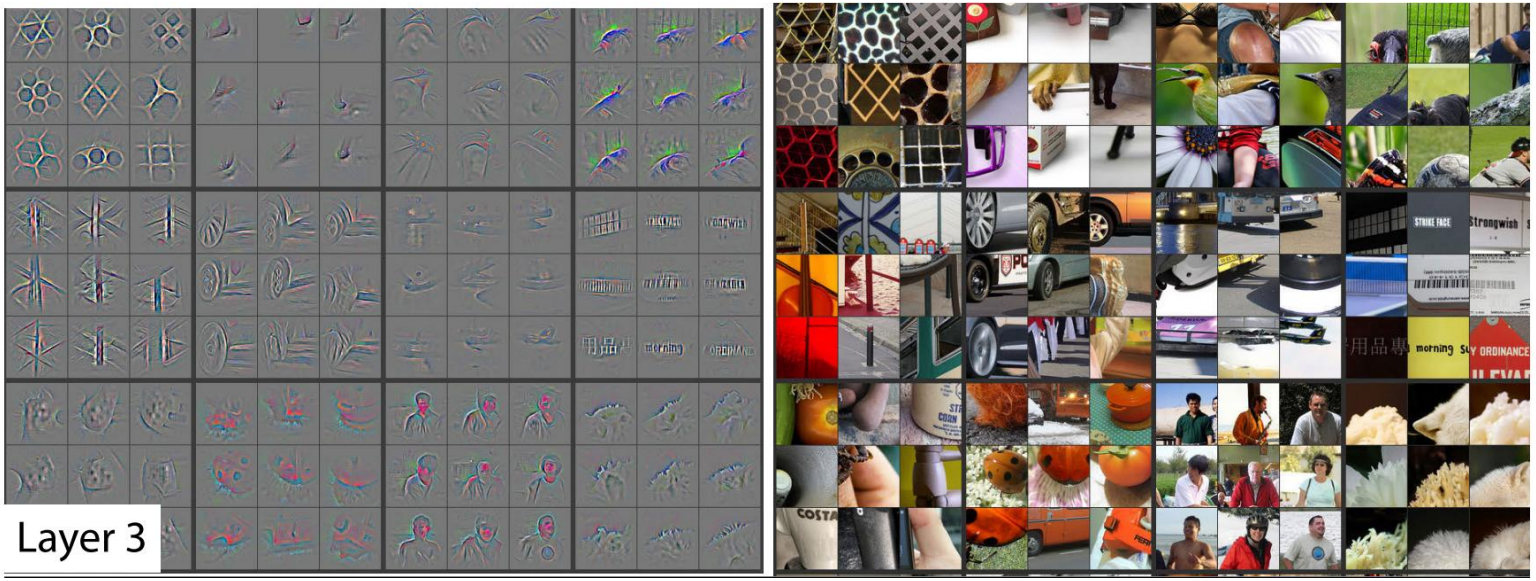


Visualizing and Understanding Convolutional Networks (ECCV 2014)

深度学习的计算图

南京大学 NANJING UNIVERSITY

隐藏层

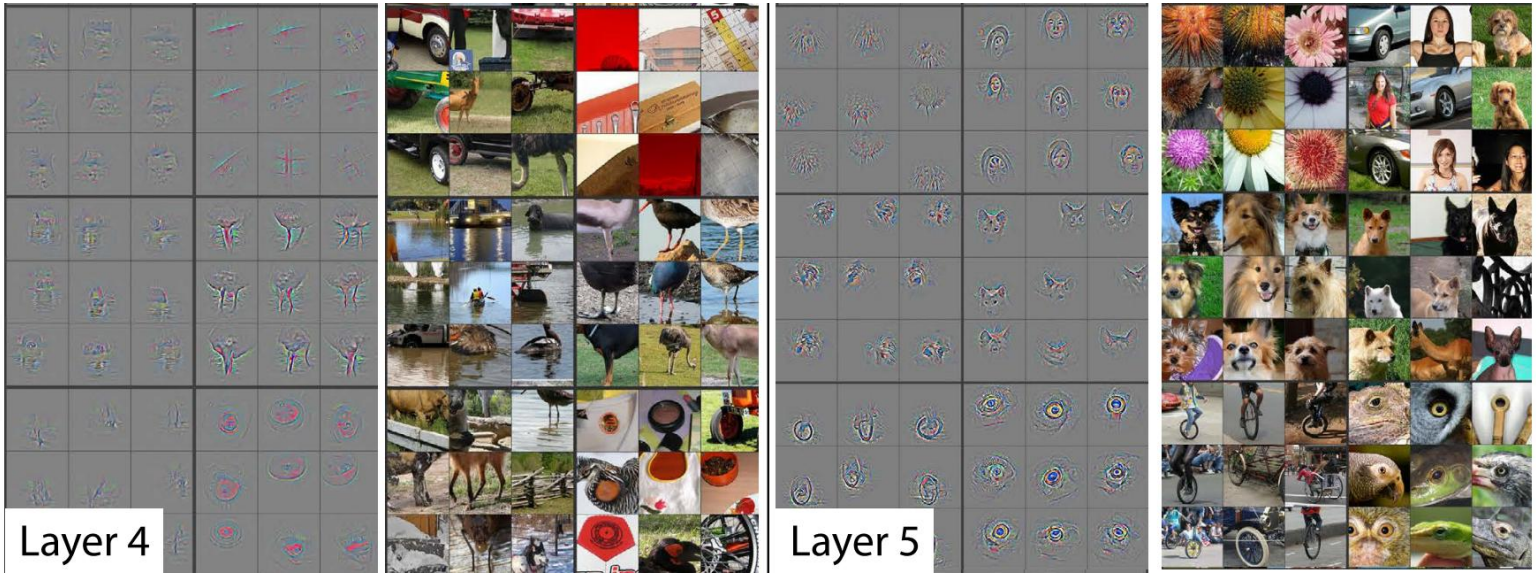


Visualizing and Understanding Convolutional Networks (ECCV 2014)

深度学习的计算图

南京大学 NANJING UNIVERSITY

隐藏层



Visualizing and Understanding Convolutional Networks (ECCV 2014)

卷积网络

卷积神经网络是一种包含空间局部连接的网络，并且在每一层中，不同单元之间权重是相同的。

核（*kernel*）：单元的权重

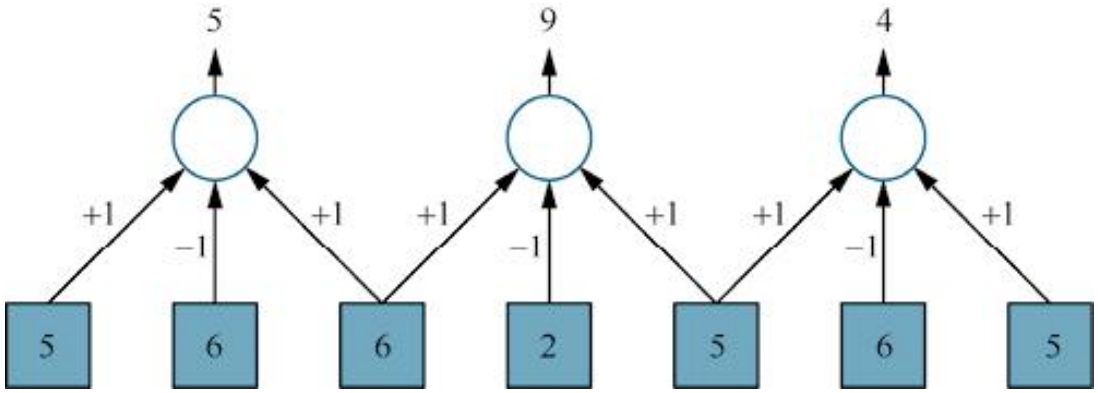
卷积：将核应用于网络的输入层或隐藏层表示的过程称为卷积

- 假设输入向量$\mathbf{x}$的大小为n，它对应于一维图像中的n个像素，核向量$\mathbf{k}$的大小为l
- 卷积运算，$z = \mathbf{x} * \mathbf{k}$.

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

- 卷积核中心之间的间隔距离称为步长s
- 卷积过程在图像边缘停止，可以用额外的像素对输入进行扩充，这样一来，卷积核就可以精确地被应用$\lfloor n/s \rfloor$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（未在图中给出）

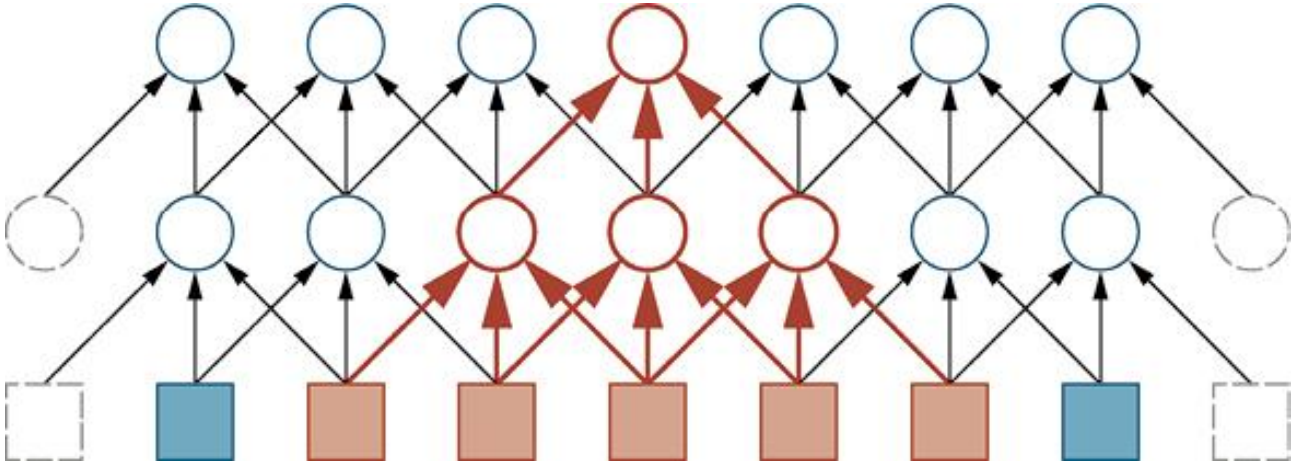
卷积网络

卷积操作可以看作如下的矩阵乘法：

在这个权重矩阵中，核出现在每一行，并按照步长相对于前一行进行移动

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

卷积网络



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

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

两种池化的常见形式:

- **平均池化（*average-pooling*）**：计算l 个输入的平均值
- **最大池化（max-pooling）:**计算l 个输入的最大值

卷积神经网络中的张量运算

- 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
- 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
- 张量运算的计算效率：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

残差网络

- **残差网络**用于构造深层网络且同时避免梯度消失问题
- 典型的深度模型：每一层将前一层的表示完全取代
- 残差网络的核心思想：每一层应当对前一层的表示进行扰动，而不是完全替换它

$$z^{(i)} = \mathbf{g}^{(i)}(\mathbf{z}^{(i-1)} + f(\mathbf{z}^{(i-1)}))$$

$\mathbf{z}^{(i)}$: 第i层的单元值, $\mathbf{g}_r$: 残差层的激活函数
f 表示残差，对从第$i-1$层传递到第i层的信息进行扰动

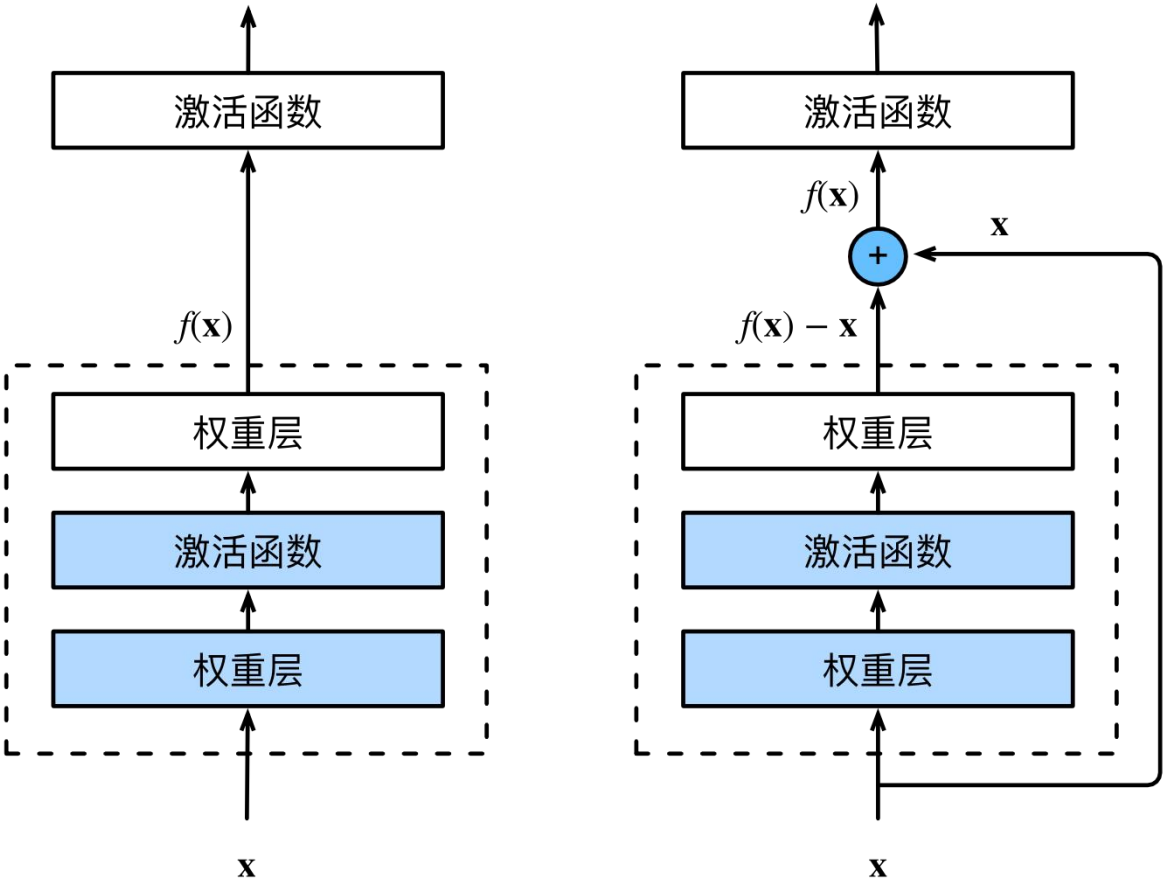
- 通常选择带有一个非线性层与一个线性层的神经网络作为计算残差的函数:

$$f(\mathbf{z}) = \mathbf{V}\mathbf{g}(\mathbf{W}\mathbf{z})$$

其中$\mathbf{W}$和$\mathbf{V}$ 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



一个正常块（左图）和一个残差块（右图）

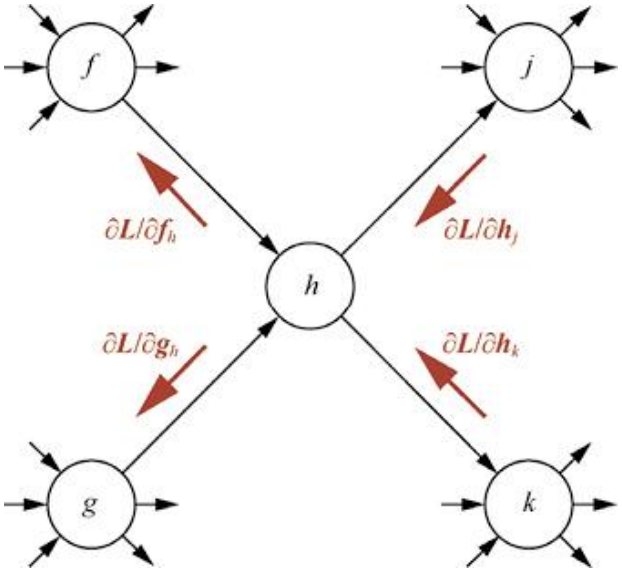
学习算法

南京大学 NANJING UNIVERSITY

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
- 这些消息均为损失函数L的偏导数：

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



任意计算图中梯度信息反向传播的图示。网络输出的前向计算从左到右进行，梯度的反向传播从右到左进行。

- 反向传播缺点：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。

学习算法

南京大学 NANJING UNIVERSITY

批量归一化

- 提高SGD的收敛速度
- 对每个小批量样例在网络内部层生成的值进行重新缩放

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

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。

如果某一层的权重非常小，并且该层的标准差减小至接近0，那么如果没有批量归一化，该层的信息就会丢失。

泛化

南京大学 NANJING UNIVERSITY

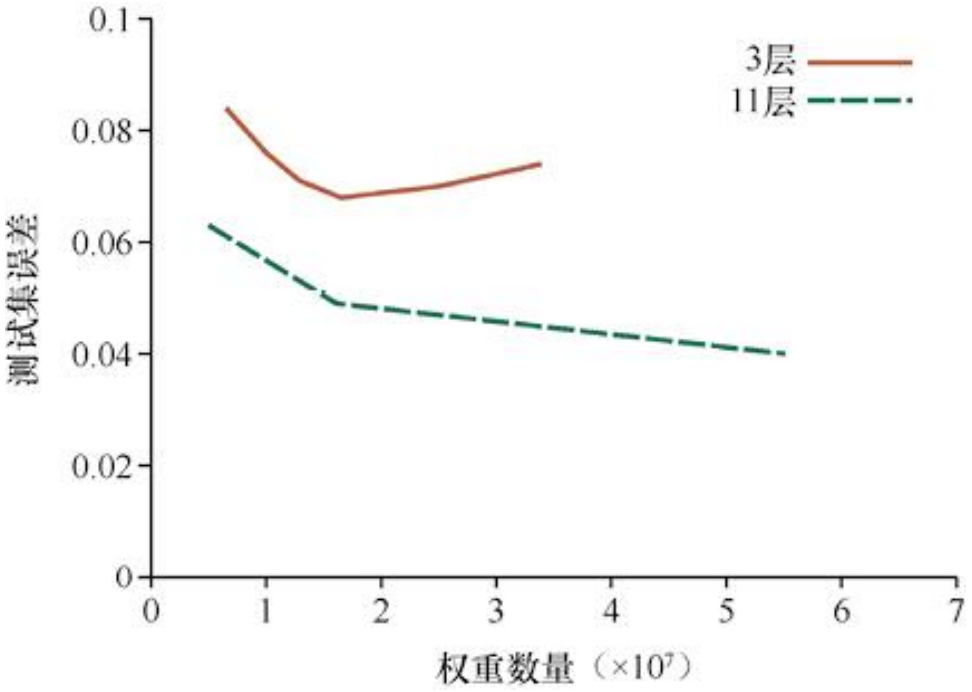
选择网络架构

- 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
- 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
- 对于具有高维输入的图像、视频、语音信号等任务，深度学习的性能优于任何其他纯粹的机器学习方法
- 深度学习模型是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
- 可能产生某些不直观的错误。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

南京大学 NANJING UNIVERSITY

选择网络架构



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

- 探索可能的网络架构的状态空间
- 进化算法：既可以对网络进行重组（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
- 主要的挑战是估计候选网络的价值
- 可以训练一个大的网络，然后搜索网络中性能更好的子图
- 启发式评价函数

泛化

南京大学 NANJING UNIVERSITY

权重衰减

- 权重衰减（*weight decay*）类似于正则化（限制模型的复杂性）
- 对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

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

第一项是交叉熵损失；第二项倾向于选择先验分布下可能性较高的权重

- 零均值高斯先验：$\log P(\boldsymbol{W}) = -\lambda \sum_{i,j} W_{i,j}^2$

泛化

暂退法（Dropout）

通过干预网络以减少测试集误差，其代价是使得网络更难拟合训练集：

- 通过在训练时引入噪声，这将迫使模型对噪声具有健壮性
- 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
- 暂退法近似了精简网络的大规模集成
- 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

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

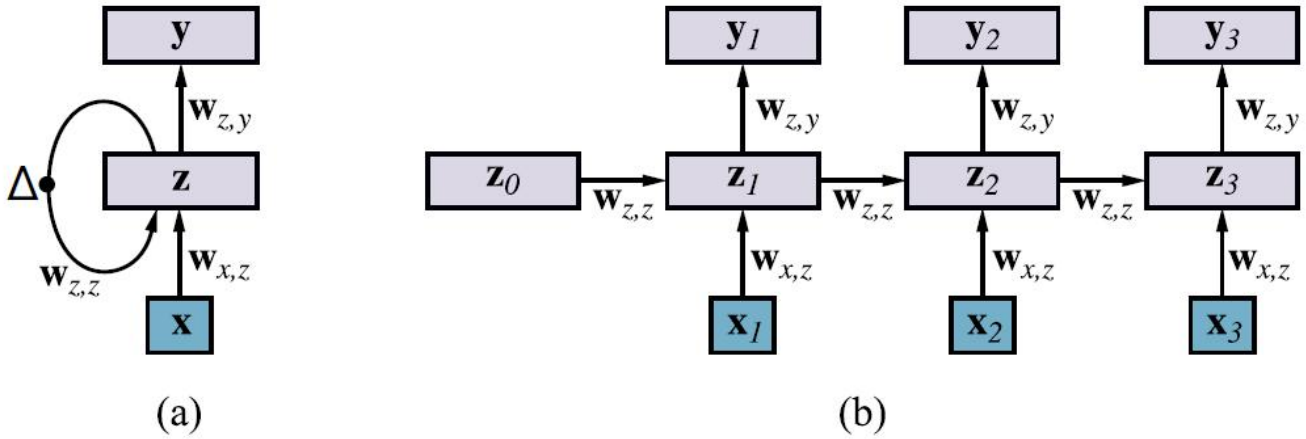
- 循环神经网络（recurrent neural network, RNN）允许计算图中存在环
- 每个环有一个延迟
  - 单元可以把在较早的时间步中所得的输出作为输入
  - RNN具有内部状态或记忆（memory）
  - 与前馈网络相比，RNN具有更强的表达能力，可以学习到数据中的长距离依赖关系

循环神经网络

训练基本的RNN

- 输入层 **x**, 具有循环连接的隐藏层 **z**, 以及输出层 **y**
- $\mathbf{g}_z$ 和 $\mathbf{g}_y$分别表示隐藏层和输出层的激活函数.

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



（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 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

南京大学 NANJING UNIVERSITY

训练基本的RNN

隐藏单元z_t的梯度可以从之前的时间步中获得：

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

$$= g_z'(in_{z,t}) \left(z_{t-1} + w_{z,z} \frac{\partial z_{t-1}}{\partial w_{z,z}}\right),$$

梯度表达式是递归的

基于时间的反向传播（*Backpropagation through time, BPTT*）：

- 计算时间步t对梯度的贡献要用到时间步$t-1$的贡献$t-1$.
- 梯度计算的总运行时间将与网络的大小呈线性关系

对于激活函数 sigmoids, tanhs, 和 ReLUs的导数小于等于1，如果 $w_{z,z} < 1$，
简单RNN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

长短期记忆循环神经网络（Long short-term memory , LSTM）

记忆单元c，存储长期记忆，从一个时间步被复制到另一个时间步。

门单元：
- 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

门单元的类型:
- **遗忘门f** 决定了记忆单元中的每个元素是否被记住
- **输入门i** 决定了记忆单元中的每个元素是否被加性更新
- **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**

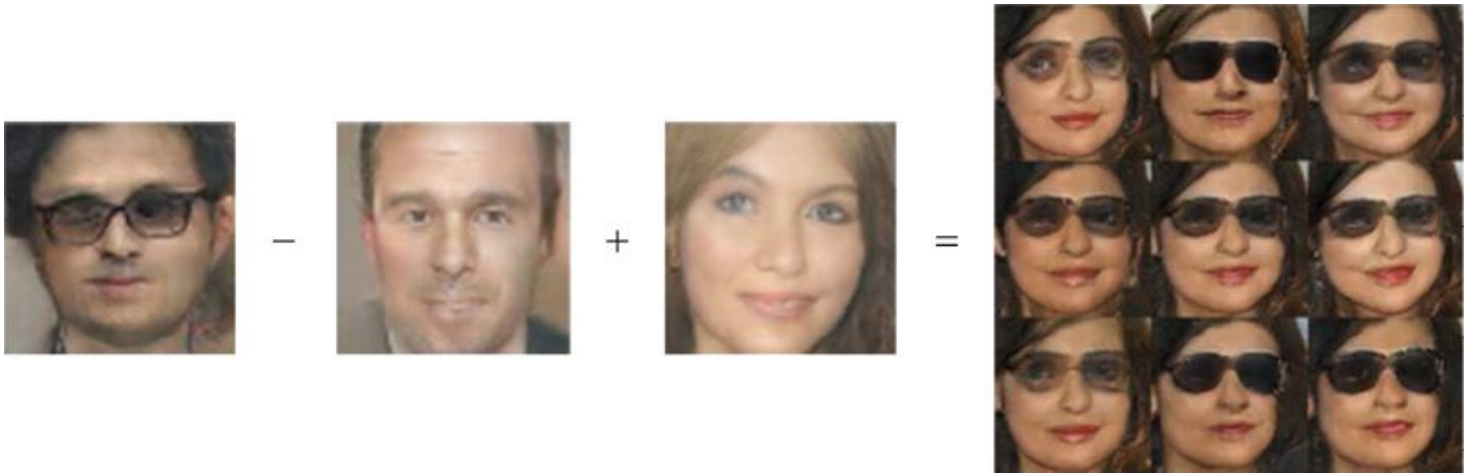
LSTM中门单元的值在[0, 1]范围内，并且是当前输入和先前隐藏状态经过sigmoid函数而获得的输出：

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

无监督学习

无监督学习算法使用一组无标签样例$\boldsymbol{x}$的训练集

- **学习新的表示**
- **学习一个生成模型**



生成模型如何使用$\boldsymbol{z}$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$\boldsymbol{z}$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$\boldsymbol{z}$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）

概率主成分分析：一个简单的生成模型

- **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

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

可以通过梯度方法或者高效的*EM*迭代算法进行求解

自编码器

自编码器包含两个部分:

- 编码器把观测数据 $\mathbf{x}$ 映射到表示 $\hat{\mathbf{z}}$
- 解码器把表示 $\hat{\mathbf{z}}$ 映射到观测数据 $\mathbf{x}$

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

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

变分自编码器(*VAE*) 更加复杂能够学到更复杂的生成模型

- 使用**变分后验** $Q(\mathbf{z})$，作为真实后验分布的近似值
- *KL*散度: “尽可能接近”

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

- **变分下界** $\mathcal{L}$, 有时也称**证据下界**（evidence lower bound, ELBO）：

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

自编码器

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

变分学习涉及关于P和Q的参数最大化$\mathcal{L}$的过程

解码器g(z)可以解释为 log P(**x** | **z**).

自回归模型:

- 向量**x**的每个元素x_i是基于向量其他元素进行预测得到的，并且不含有隐变量
- 如果**x**的大小是固定的，那么自回归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
- 常用于时序数据分析

深度自回归模型:

- 线性高斯模型被替换为具有适当输出层的任意深度网络，其具体形式取决于x_t是离散的还是连续的

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

一对网络结合在一起形成的生成模型

- **生成器**，将$\boldsymbol{z}$ 映射到 $\boldsymbol{x}$
- **判别器**，一个分类器，被训练来判断输入的$\boldsymbol{x}$为真（从训练集中获取的）或假（由生成器生成的）

GAN是一种**隐式模型**：即样本可以被生成，但其概率不易获得

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

GAN在图像生成任务中已经取得了很大成功。例如，GAN可以创造一幅关于某个不存在的人的十分逼真的、高分辨率的图像

无监督学习与迁移学习

南京大学 NANJING UNIVERSITY

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

StyleGAN (CVPR 2019)

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

冻结预训练模型的前几层，因为这些层起到了特征检测器的作用

将只修改较高层的参数
- 识别特定问题的特征并对其进行分类

对于自然语言处理，目前常用的做法是从一个预先训练好的模型出发，以两个方式对模型进行微调：
- 给出所需领域中使用的专业词汇的样本
- 在所需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。

小结

- **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。