一、神经网络基础

1、神经网络中的激活函数：对比ReLU与Sigmoid、Tanh的优缺点？ReLU有哪些变种？

优点：

从计算的角度上，Sigmoid和Tanh激活函数均需要计算指数，复杂度高，而ReLU只需要一个阈值即可得到激活值；
ReLU的非饱和性可以有效地解决梯度消失的问题，提供相对宽的激活边界。
ReLU的单侧抑制提供了网络的稀疏表达能力。

> 修正线性单元（Rectified Linear Unit，ReLU）：ReLU函数被认为有生物上的解释性，比如单侧抑制、宽兴奋边界（即兴奋程度 也可以非常高）。在生物神经网络中，同时处于兴奋状态的神经元非常稀疏。人 脑中在同一时刻大概只有1 ~ 4%的神经元处于活跃状态。Sigmoid型激活函数会导致一个非稀疏的神经网络，而ReLU却具有很好的稀疏性，大约50%的神 经元会处于激活状态。

缺点：

ReLU和Sigmoid一样，它们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。

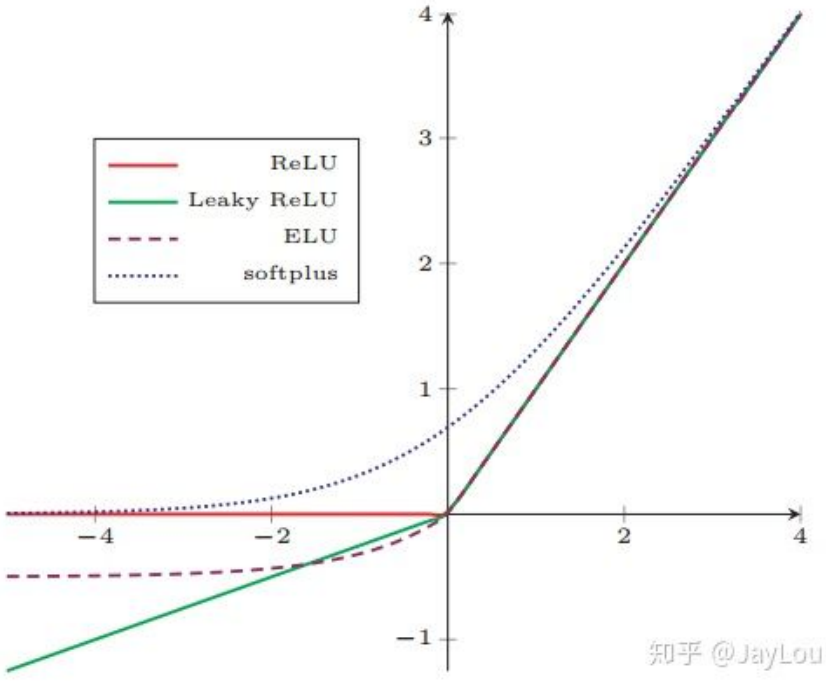
> ***Sigmoid函数的输出不是零中心的***。这个性质并不是我们想要的，因为在神经网络后面层中的神经元得到的数据将不是零中心的。这一情况将影响梯度下降的运作，因为如果输入神经元的数据总是正数（比如在 $f = w^T x + b$ 中每个元素都 $x > 0$ ），那么关于 w 的梯度在反向传播的过程中，将会要么全部是正数，要么全部是负数（具体依整个表达式 f 而定）。这将会导致梯度下降权重更新时出现z字型的下降。然而，可以看到整个批量的数据的梯度被加起来后，对于权重的最终更新将会有不同的正负，这样就从一定程度上减轻了这个问题。因此，该问题相对于上面的神经元饱和问题来说只是个小麻烦，没有那么严重。

ReLU的局限性在于其训练过程中会导致神经元死亡的问题。

> 在训练时，如果参数在一次不恰当的更新后，第一个隐藏层中的某个ReLU神经元在所有的训练数据上都不能被激活。那么，这个神经元自身参数的梯度永远都会是 0，在以后的训练过程中永远不能被激活。这种现象称为死亡 ReLU 问题 （ Dying ReLU Problem ），并且也有可能会发生在其它隐藏层。

ReLU的变种

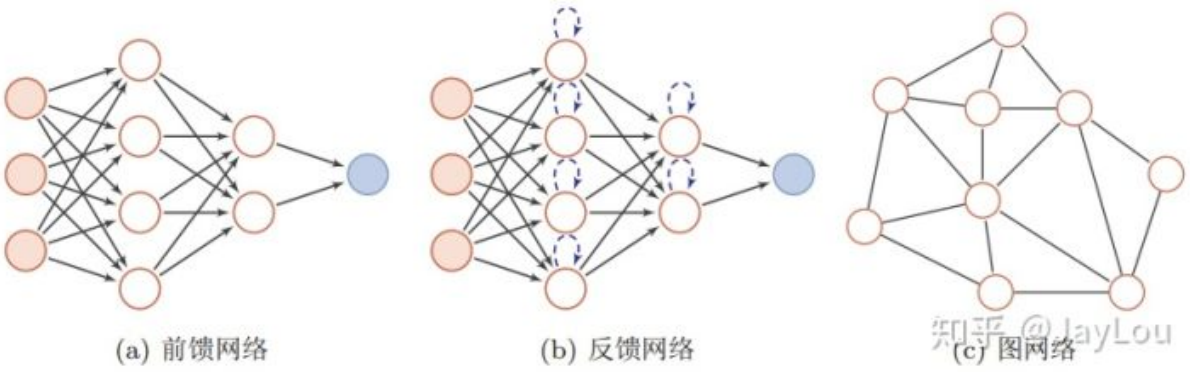
在实际使用中，为了避免上述情况，有几种ReLU的变种也会被广泛使用：Leaky ReLU、ELU以及softplus函数



ReLU、Leaky ReLU、ELU以及softplus函数

2、神经网络结构哪几种？各自都有什么特点？

前馈网络：整个网络中的信息是朝一个方向传播，没有反向的信息传播，可以用一个**有向无环路图**表示。前馈网络包括**全连接前馈网络**和**卷积神经网络**等。
反馈网络：反馈网络中神经元不但可以接收其它神经元的信号，也可以接收自己的反馈信号。和前馈网络相比，反馈网络中的神经元具有记忆功能，在不同的时刻具有不同的状态。馈神经网络中的信息传播可以是单向或双向传递，因此可用一个**有向循环图或无向图**来表示。反馈网络包括**循环神经网络、Hopfield 网络、玻尔兹曼机**等。
图网络：图网络是定义在图结构数据上的神经网络。图中每个节点都一个或一组神经元构成。节点之间的连接可以是有向的，也可以是无向的。每个 节点可以收到来自相邻节点或自身的信息。 图网络是前馈网络和记忆网络的泛化，包含很多不同的实现方式，比如图卷积网络、消息传递网络等。



前馈网络、反馈网络和图网络

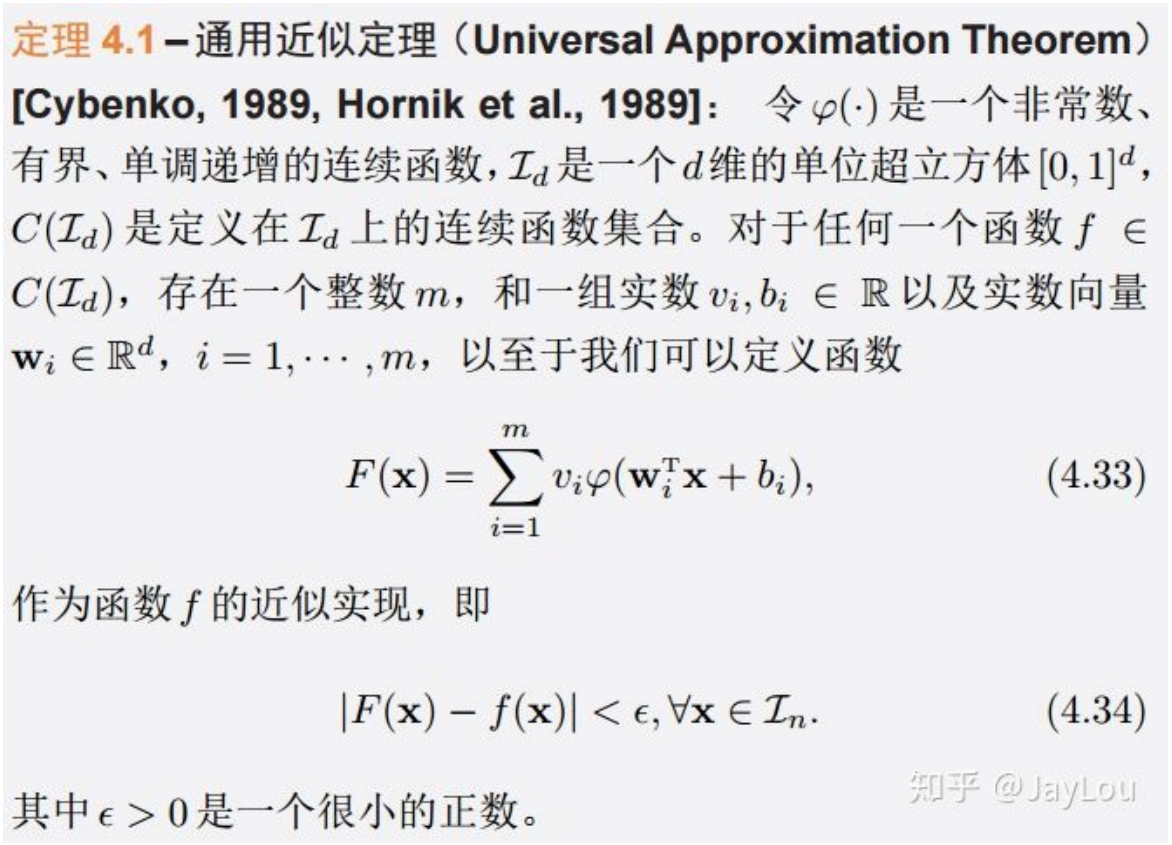
3、前馈神经网络叫做多层感知机是否合适？

前馈神经网络也经常称为多层感知器（Multi-Layer Perceptron，MLP）。但多层感知器的叫法并不是十分合理，因为前馈神经网络其实是由多层的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。

4、前馈神经网络怎么划分层？

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受前一层神经元的信号，并产生信号输出到下一层。第0层叫输入层，最后一层叫输出层，其它中间层叫做隐藏层。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。

5、如何理解通用近似定理？



定理 4.1 – 通用近似定理（Universal Approximation Theorem）[Cybenko, 1989, Hornik et al., 1989]： 令 $\varphi(\cdot)$ 是一个非常数、有界、单调递增的连续函数，$\mathcal{I}_d$ 是一个 d 维的单位超立方体 $[0,1]^d$，$C(\mathcal{I}_d)$ 是定义在 $\mathcal{I}_d$ 上的连续函数集合。对于任何一个函数 $f \in C(\mathcal{I}_d)$，存在一个整数 m，和一组实数 $v_i, b_i \in \mathbb{R}$ 以及实数向量 $\mathbf{w}_i \in \mathbb{R}^d$，$i = 1, \cdots, m$，以至于我们可以定义函数

$$F(\mathbf{x}) = \sum_{i=1}^{m} v_i \varphi(\mathbf{w}_i^{\mathrm{T}} \mathbf{x} + b_i), \tag{4.33}$$

作为函数 f 的近似实现，即

$$|F(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \forall \mathbf{x} \in \mathcal{I}_n. \tag{4.34}$$

其中 $\epsilon > 0$ 是一个很小的正数。

通用近似定理《神经网络与深度学习》

通用近似定理：对于具有线性输出层和**至少一个使用“挤压”性质的激活函数的隐藏层**组成的前馈神经网络，只要其隐藏层神经元的数量足够，它可以以任意的精度来近似任何从一个定义在实数空间中的有界闭集函数。说明：

所谓“挤压”性质的函数是指像 sigmoid 函数的有界函数，但神经网络的通用近似性质也被证明对于其 它类型的激活函数，比如ReLU，也都是适用的。
通用近似定理只是说明了神经网络的计算能力可以去近似一个给定的连续函数，但并没有给出如何找到这样一个网络，以及是否是最优的。此外，当应用到机器学习时，真实的映射函数并不知道，一般是通过经验风险最小化和正则化来进行参数学习。因为神经网络的强大能力，反而容易在训练集上过拟合。

6、怎么理解前馈神经网络中的反向传播？具体计算流程是怎样的？

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial W_{ij}^{(l)}} = \left(\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} \right)^{\mathrm{T}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}};$$

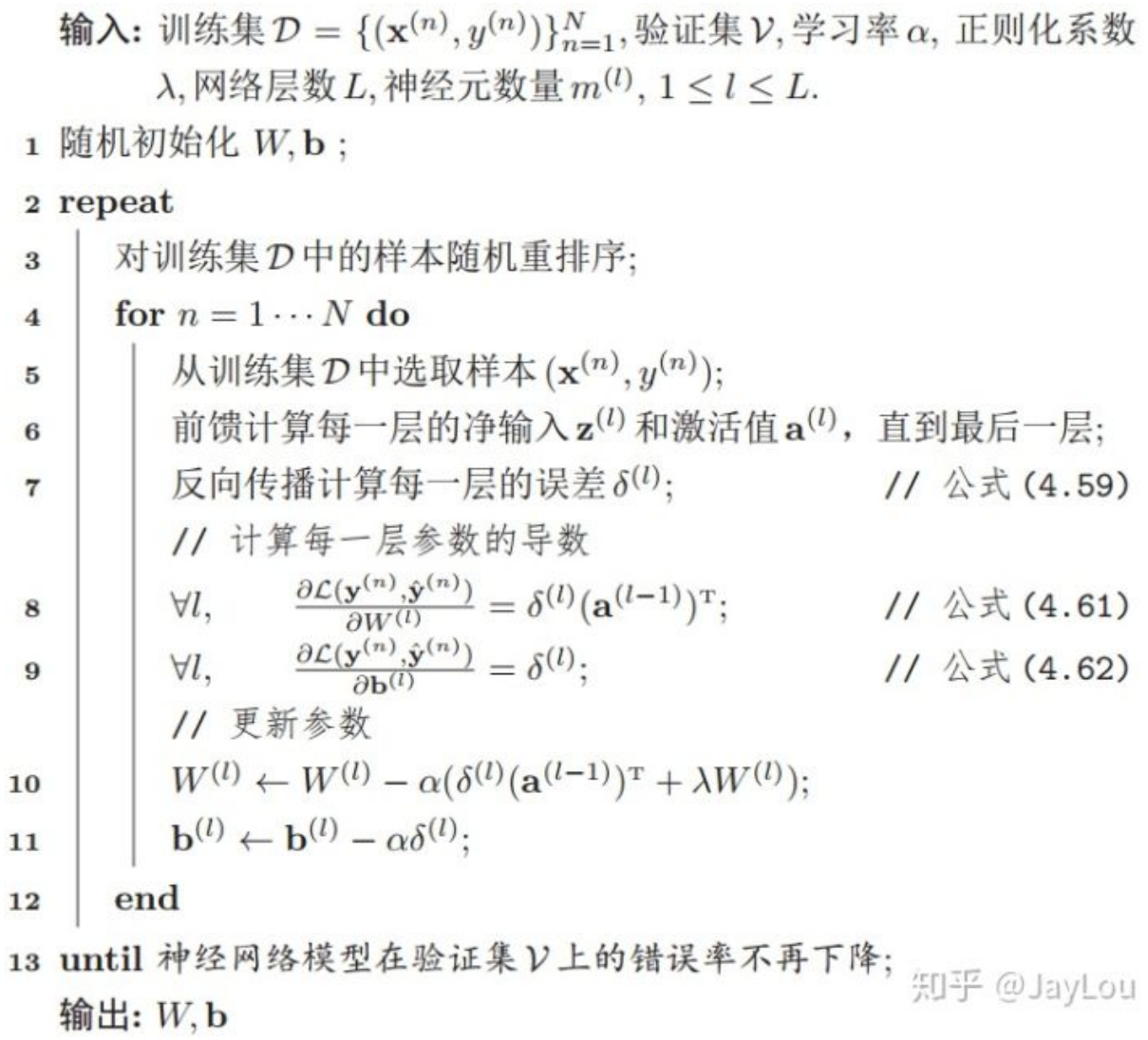
$$
\begin{aligned}
\delta^{(l)} &\triangleq \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \\
&= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(\mathbf{y},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

上式中，误差项 $\delta^{(l)}$ 来表示第 l 层的神经元对最终误差的影响，也反映了最终的输出对第 l 层的神经元对最终误差的敏感程度。

前馈神经网络中反向传播的实质就是误差的反向传播： 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。

误差反向传播算法的具体含义是：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。

出使用随机梯度下降的误差反向传播算法的具体训练过程：



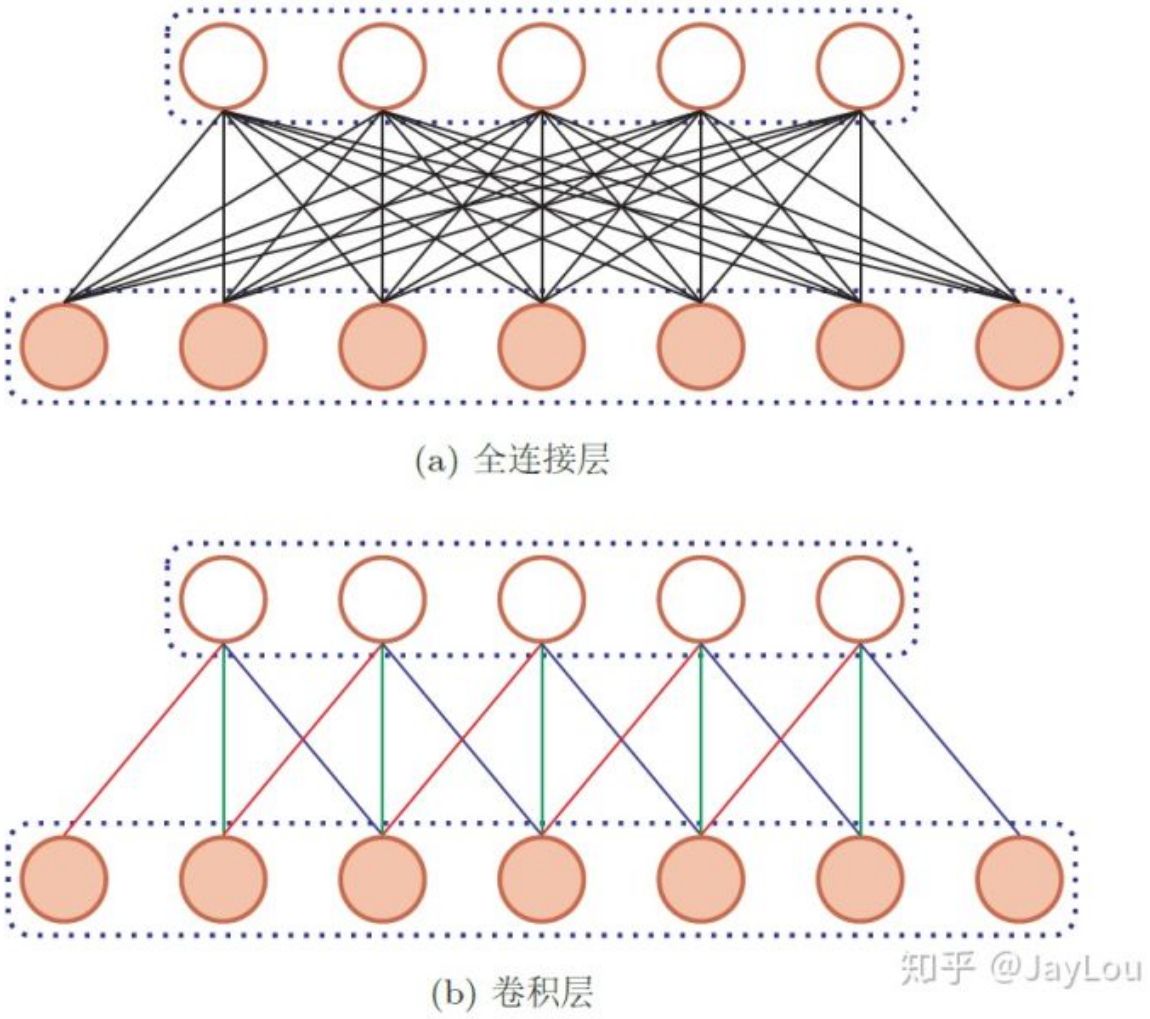
7、卷积神经网络哪些部分构成？各部分作用分别是什么？

如果用全连接前馈网络来处理图像时，会存在以下两个问题：

参数太多；
局部不变性特征：全连接前馈网络很难提取局部不变特征，一般需要进行数据增强来提高性能。

卷积神经网络一般是由卷积层、汇聚层和全连接层交叉堆叠而成的前馈神经网络，使用反向传播算法进行训练。卷积神经网络有三个结构上的特性：局部连接，权重共享以及子采样。

卷积层的作用：局部连接，权重共享；
池化层（pooling layer）也叫子采样层（subsampling layer）的作用：进行特征选择，降低特征数量，并从而减少参数数



全连接和局部连接

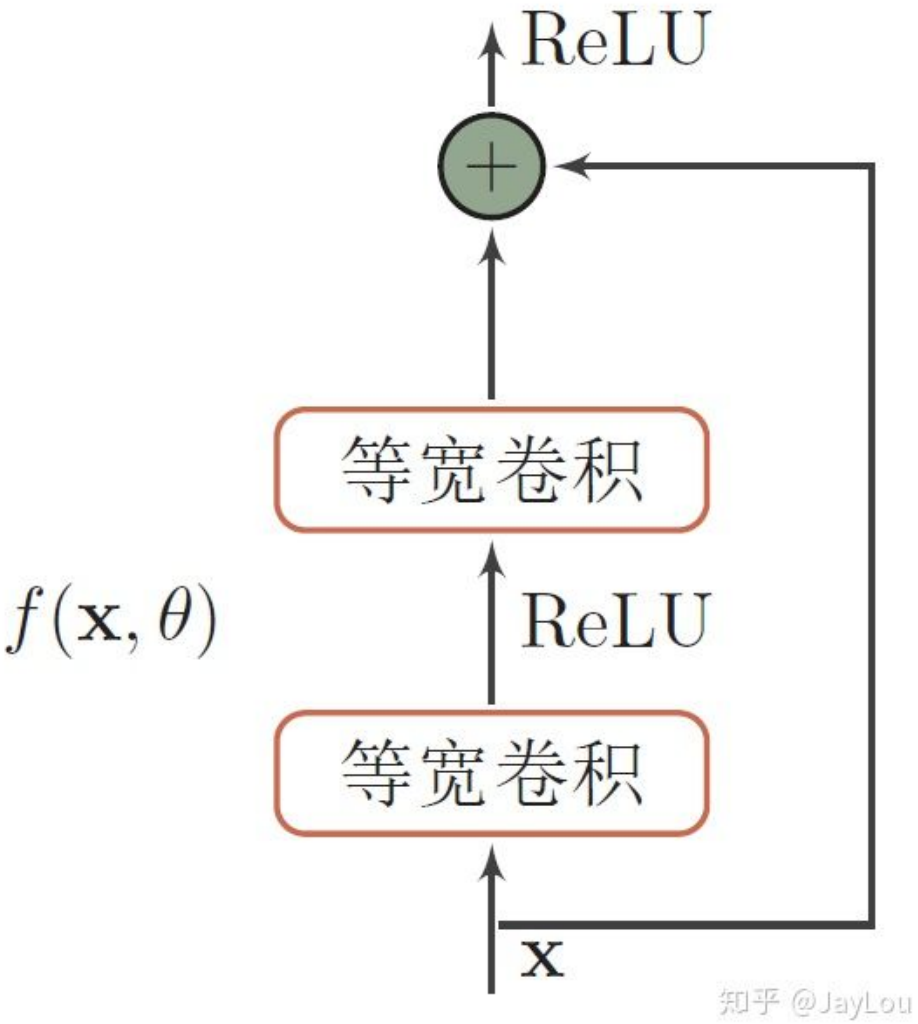
8、在深度学习中，网络层数增多会伴随哪些问题，怎么解决？为什么要采取残差网络ResNet？

CNN之三大经典网络LeNet-AlexNet-VGG。在VGG中，卷积网络达到了19层，在GoogLeNet中，网络史无前例的达到了22层。那么，网络的精度会随着网络的层数增多而增多吗？随着网络层数的增加，网络发生了退化（**degradation**）的现象：随着网络层数的增多，训练集loss逐渐下降，然后趋于饱和，当你再增加网络深度的话，训练集loss反而会增大。注意这并不是过拟合，因为在过拟合中训练loss是一直减小的。

（1）在深度学习中，网络层数增多会伴随哪些问题，怎么解决？

1 计算资源的消耗（GPU）
2 模型容易过拟合（Dropout）
3 梯度消失/梯度爆炸问题的产生（批量归一化BN）：BN层能对各层的输出做归一化，这样梯度在反向层层传递后仍能保持大小稳定，不会出现过小或过大的情况。
4 degradation退化问题：随着网络层数的增多，训练集loss逐渐下降，然后趋于饱和，当你再增加网络深度的话，训练集loss反而会增大。注意这并不是过拟合，因为在过拟合中训练loss是一直减小的。（残差网络ResNet？）

（2）为什么要采取残差网络ResNet？



残差网络ResNet

假设在一个深度网络中，我们期望一个非线性单元（可以为一层或多层的卷积层）$f(\mathbf{x},\theta)$去逼近一个目标函数为$h(\mathbf{x})$。如果将目标函数拆分成两部分：恒等函数（identity function）$\mathbf{x}$和残差函数（residue function）$h(\mathbf{x})-\mathbf{x}$两个部分，

$$h(\mathbf{x}) = \underbrace{\mathbf{x}}_{\text{恒等函数}} + \underbrace{\big(h(\mathbf{x})-\mathbf{x}\big)}_{\text{残差函数}}. \tag{5.38}$$

根据通用近似定理，一个由神经网络构成的非线性单元有足够的能力来近似逼近原始目标函数或残差函数，但实际中后者更容易学习[He et al., 2016]。因此，原来的优化问题可以转换为：让非线性单元$f(\mathbf{x},\theta)$去近似残差函数$h(\mathbf{x})-\mathbf{x}$，并用$f(\mathbf{x},\theta)+\mathbf{x}$去逼近$h(\mathbf{x})$。

残差块可以表示为：（参考自详解残差网络）

$$x_{l+1} = x_l + \mathcal{F}(x_l, W_l)$$

对于一个更深的层 L ，其与 l 层的关系可以表示为

$$x_L = x_l + \sum_{i=1}^{L-1} \mathcal{F}(x_i, W_i)$$

这个公式反应了残差网络的两个属性：

L 层可以表示为任意一个比它浅的l层和他们之间的残差部分之和；

$x_L = x_0 + \sum_{i=0}^{L-1} \mathcal{F}(x_i, W_i)$ ， L 是各个残差块特征的单位累和，而MLP是特征矩阵的累积。

根据BP中使用的导数的链式法则，损失函数 ε 关于 x_l 的梯度可以表示为

$$\frac{\partial \varepsilon}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1}\mathcal{F}(x_i, W_i)) = \frac{\partial \varepsilon}{\partial x_L} + \frac{\partial \varepsilon}{\partial x_L}\frac{\partial}{\partial x_l}\sum_{i=1}^{L-1}\mathcal{F}(x_i, W_i)$$

上面公式反映了残差网络的两个属性：

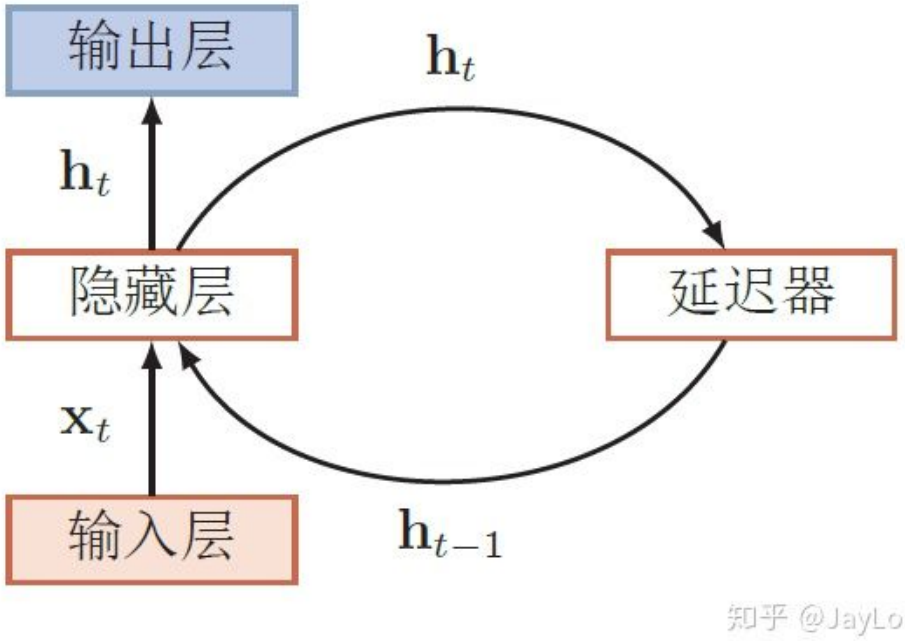
在整个训练过程中， $\frac{\partial}{\partial x_l}\sum_{i=1}^{L-1}\mathcal{F}(x_i, W_i)$ 不可能一直为 -1 ，也就是说在残差网络中不会出现梯度消失的问题。

$\frac{\partial \varepsilon}{\partial x_L}$ 表示 L 层的梯度可以直接传递到任何一个比它浅的 l 层。

二、循环神经网络

1、什么是循环神经网络？循环神经网络的基本结构是怎样的？

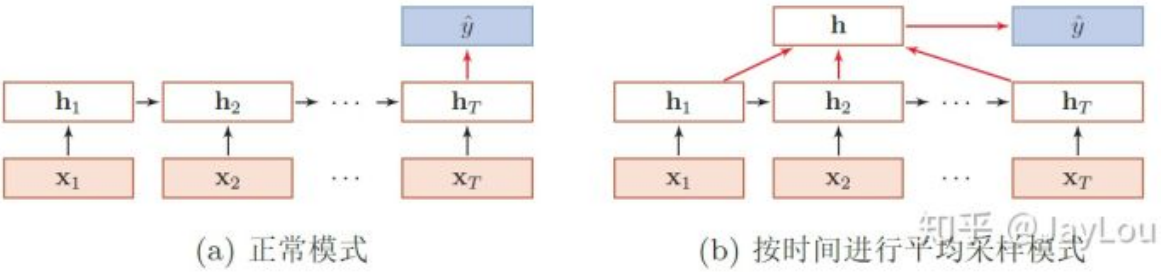
> 循环神经网络（Recurrent Neural Network，RNN）是一类具有短期记忆能力的神经网络。在循环神经网络中，神经元不但可以接受其它神经元的信息，也可以接受自身的信息，形成具有环路的网络结构。和前馈神经网络相比，循环神经网络更加符合生物神经网络的结构。循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上。循环神经网络的参数学习可以通过随时间反向传播算法来学习。随时间反向传播算法即按照时间的逆序将错误信息一步步地往前传递。当输入序列比较长时，会存在梯度爆炸和消失问题，也称为长期依赖问题。为了解决这个问题，人们对循环神经网络，进行了很多的改进，其中最有效的改进方式引入门控机制。此外，循环神经网络可以很容易地扩展到两种更广义的记忆网络模型：递归神经网络和图网络。
> ——《神经网络与深度学习》



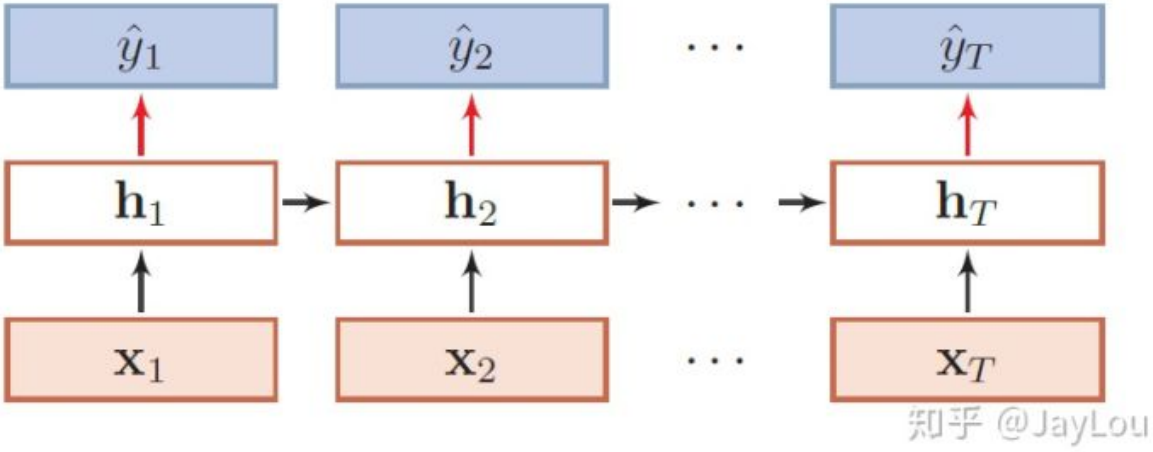
循环神经网络

2、循环神经网络RNN常见的几种设计模式是怎样的？

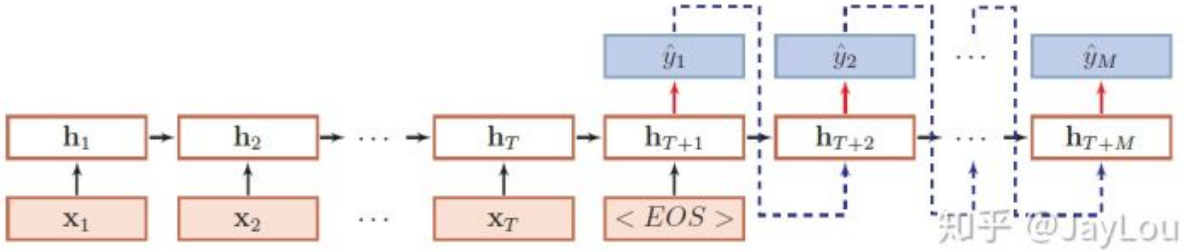
序列到类别模式
同步序列到序列模式
异步的序列到序列模式



序列到类别模式



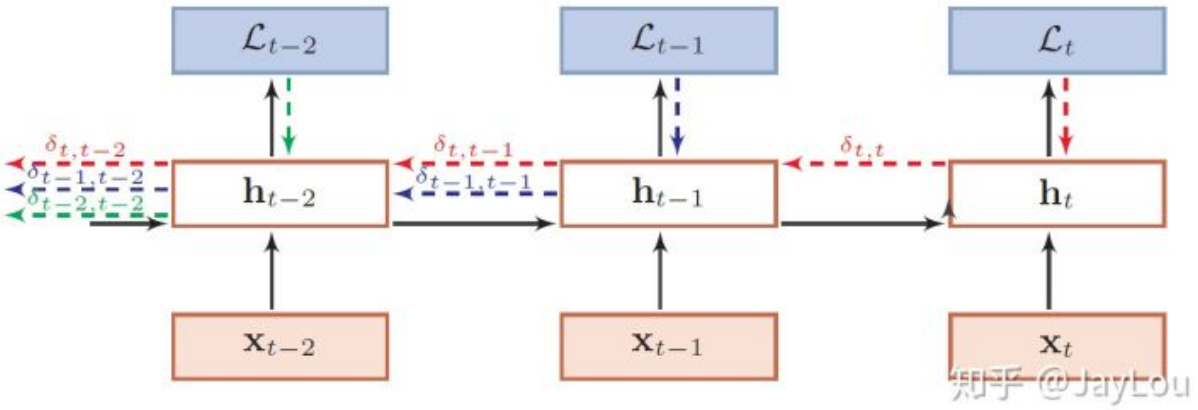
同步序列到序列模式



异步的序列到序列模式

3、循环神经网络RNN怎样进行参数学习？

随时间反向传播（Backpropagation Through Time，**BPTT**）算法的主要思想是通过类似前馈神经网络的错误反向传播算法来进行计算梯度。BPTT算法将循环神经网络看作是一个展开的多层前馈网络，其中“每一层”对应循环网络中的“每个时刻”。这样，循环神经网络就可以按按照前馈网络中的反向传播算法进行计算参数梯度。**在“展开”的前馈网络中，所有层的参数是共享的。**因此参数的真实梯度是将所有“展开层”的参数梯度之和。



随时间反向传播BPTT算法示例

$$\begin{aligned}\delta_{t,k} &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \\ &= \frac{\partial \mathbf{h}_k}{\partial \mathbf{z}_k}\frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{h}_k}\frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_{k+1}} \\ &= \mathrm{diag}\big(f'(\mathbf{z}_k)\big)U^{\mathrm{T}}\delta_{t,k+1}.\end{aligned}$$

知乎 @JayLou

定义 $\delta_{t,k}$ 为第t 时刻的损失对第k 时刻隐藏神经层的净输入 $z_k = Uh_{k-1} + Wx_k + b$ 的导数，同样对U、W求导时则可得到到随时间反向传播的公式：

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}\mathbf{h}_{k-1}^{\mathrm{T}}$$

知乎 @JayLou

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T}\sum_{k=1}^{t} \delta_{t,k}\mathbf{x}_k^{\mathrm{T}},$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{t=1}^{T}\sum_{k=1}^{t} \delta_{t,k}.$$

4、循环神经网络RNN长期依赖问题产生的原因是怎样的？

RNN产生长期依赖的原因与参数学习BPTT过程有关，实质就是参数U 的更新主要靠当前时刻k 的几个相邻状态 h_k 来更新，长距离的状态对U 没有影响。将 $\delta_{t,k}$ 展开可得到：

$$\delta_{t,k} = \prod_{i=k}^{t-1}(diag(f^{'}(z_i))U^T)\delta_{t,t}$$

如果令 $\gamma \simeq ||diagf^{'}(z_i)U^T||$ ，则可得到：

$$\delta_{t,k} = \gamma^{t-k}\delta_{t,t}$$

若 $\gamma > 1$ ，当 $t-k \to \infty$ 时，造成系统不稳定，称为**梯度爆炸问题**（Gradient Exploding Problem）；相反，若 $\gamma < 1$ ，当 $t-k \to \infty$ 时会出现和深度前馈神经网络类似的**梯度消失问题**（gradient vanishing problem）。

在循环神经网络中的梯度消失不是说 $\frac{\alpha L}{\alpha U}$ 的梯度消失了，而是 $\delta_{t,k}$ 的梯度消失了当 $t-k \to \infty$ 。也就是说，参数U 的更新主要靠当前时刻k 的几个相邻状态 h_k 来更新，长距离的状态对U 没有影响。

5、RNN中为什么要采用tanh而不是ReLu作为激活函数？为什么普通的前馈网络或 CNN 中采取ReLU不会出现问题？

由 $\delta_{t,k} = \prod_{i=k}^{t-1}(diag(f^{'}(z_i))U^T)\delta_{t,t}$ 可以得到，当使用ReLU作为激活函数时，$diag(f^{'}(z_i)0 = I$ ，只要 U 不是单位矩阵，梯度还是会出现消失或者爆炸的现象。

当采用ReLU作为循环神经网络中隐含层的激活函数时，只有当 U 的取值在单位矩阵附近时才能取得比较好的效果，因此需要将 U 初始化为单位矩阵。实验证明，初始化W为单位矩阵并使用ReLU激活函数在一些应用中取得了与长短期记忆模型相似的结果，并且学习速度比长短期记忆模型更快，是一个值得尝试的小技巧。

那么为什么普通的前馈网络或 CNN 中采取ReLU不会出现梯度消失或梯度爆炸的问题呢？

类似前馈神经网络中的误差反向传播：

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

因为他们每一层的 W 不同，且在初始化时是独立同分布的，因此可以在一定程度相互抵消。即使多层之后一般也不会出现数值问题。

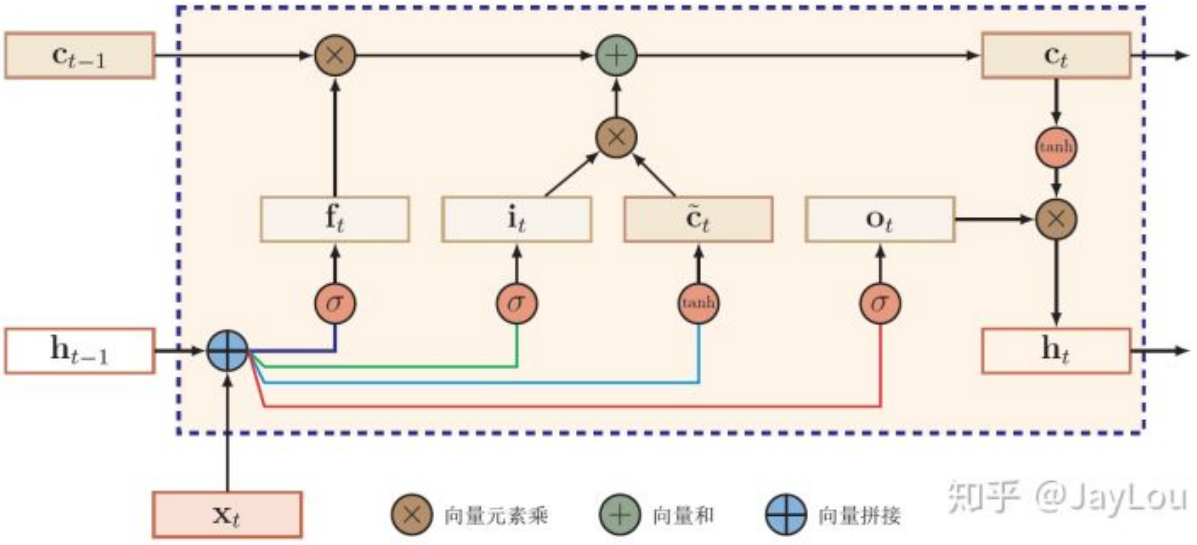
6、循环神经网络RNN怎么解决长期依赖问题？LSTM的结构是怎样的？LSTM又有哪些变种的？

RNN中的长期依赖问题，也就是梯度消失或梯度爆炸可以采取如下方法解决：

1) RNN梯度爆炸的解决方法：梯度截断

2) RNN梯度消失的解决方法；残差结构、门控机制（LSTM、GRU）

为了RNN中的长期依赖问题，一种非常好的解决方案是引入门控Hochreiter and Schmidhuber 来控制信息的累积速度，包括有选择地加入新的信息，并有选择地遗忘之前累积的信息。这一类网络可以称为基于门控的循环神经网络（Gated RNN）。



长短期记忆（LSTM）结构

长短期记忆（LSTM）网络和门控循环单元（GRU）网络是两种主要的基于门控的循环神经网络。LSTM的结构如上图所示，LSTM三个门的作用是：

- 遗忘门 $\mathbf{f}_t$ 控制上一个时刻的内部状态 $\mathbf{c}_{t-1}$ 需要遗忘多少信息。
- 输入门 $\mathbf{i}_t$ 控制当前时刻的候选状态 $\tilde{\mathbf{c}}_t$ 有多少信息需要保存。
- 输出门 $\mathbf{o}_t$ 控制当前时刻的内部状态 $\mathbf{c}_t$ 有多少信息需要输出给外部状态 $\mathbf{h}_t$。

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}; x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}; x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}; x_t] + b_C) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}; x_t] + b_o) \\
h_t &= o_t \circ \tanh(C_t)
\end{aligned}
$$

知乎 @JayLou

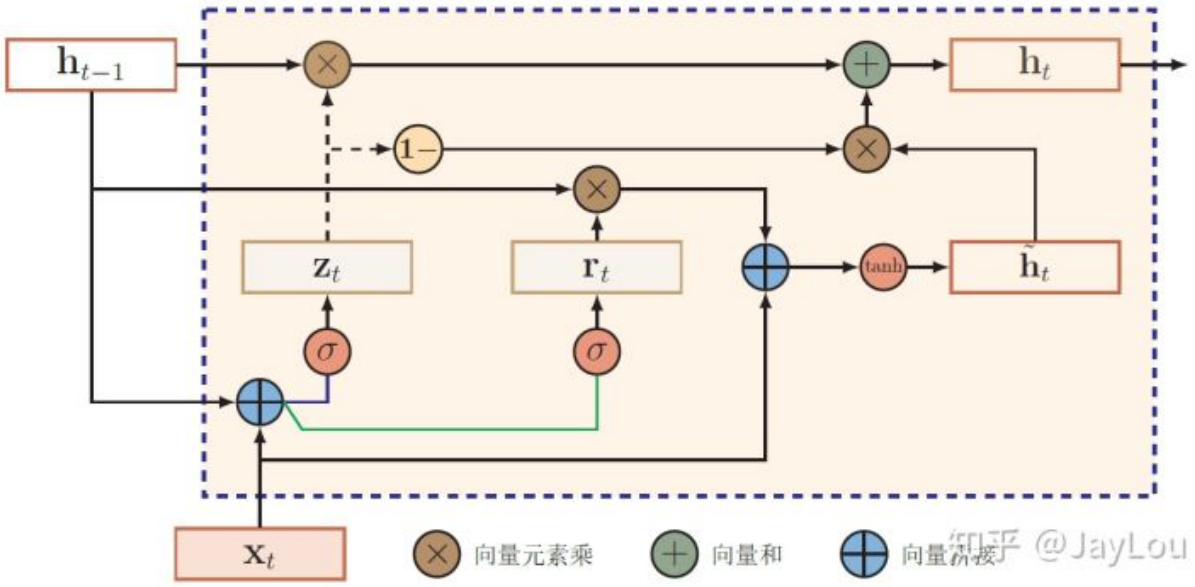
7、怎么理解“长短时记忆单元”？RNN中的隐状态 h_t 与LSTM中的记忆状态 C_t 有什么区别？

h_t 为短期记忆， C_t 记忆能力由于 h_t ，但又远远短于长期记忆，因此被称为长的短时记忆。

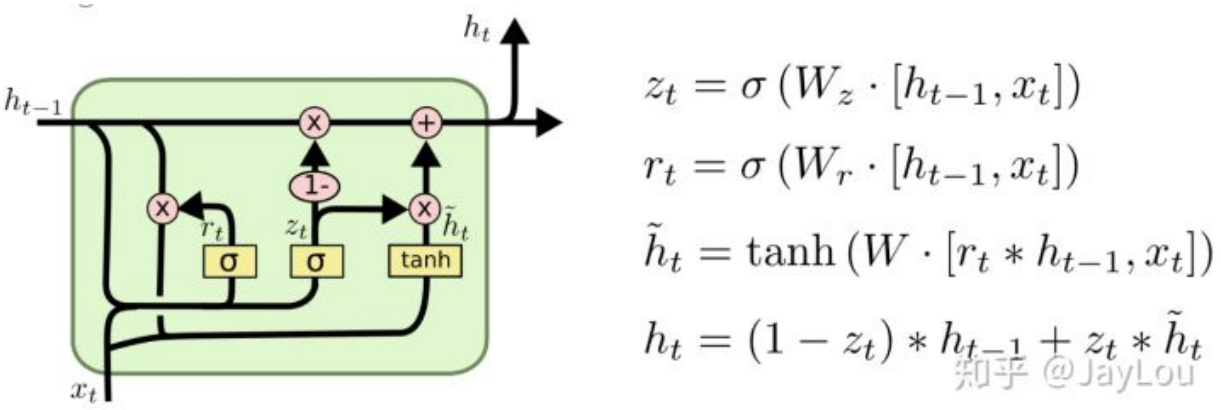
> 记忆循环神经网络中的隐状态h存储了历史信息，可以看作是一种记忆（mem-ory）。在简单循环网络中，隐状态每个时刻都会被重写，因此可以看作是一种短期记忆（short-term

> memory）。在神经网络中，长期记忆（long-term memory）可以看作是网络参数，隐含了从训练数据中学到的经验，并更新周期要远远慢于短期记忆。而在LSTM网络中，记忆单元c 可以在某个时刻捕捉到某个关键信息，并有能力将此关键信息保存一定的时间间隔。记忆单元c 中保存信息的生命周期要长于短期记忆h，但又远远短于长期记忆，因此称为长的短期记忆（long short-term memory）。

8、LSTM与GRU关系是怎样的？



GRU结构



GRU 把遗忘门和输入门合并为**更新门（update）** `z` ，并使用**重置门（reset）** `r` 代替输出门；
合并了记忆状态 `C` 和隐藏状态 `h`

三、神经网络的训练技巧及优化问题

神经网络主要的问题集中在优化问题和正则化问题。

> （1）优化问题：神经网络模型是一个非凸函数，再加上在深度网络中的梯度消失问题，很难进行优化；另外，深层神经网络模型一般参数比较多，训练数据也比较大，会导致训练的效率比较低。
> （2）泛化问题：因为神经网络的拟合能力强，反而容易在训练集上产生过拟合。因此，在训练深层神经网络时，同时也需要通过一定的正则化方法来改进网络的泛化能力。

对应的优化问题有：

1）如何初始化参数；
2）预处理数据；
3）如何避免陷入局部最优？（学习率衰减+梯度方向优化）

具体展开讨论如下：

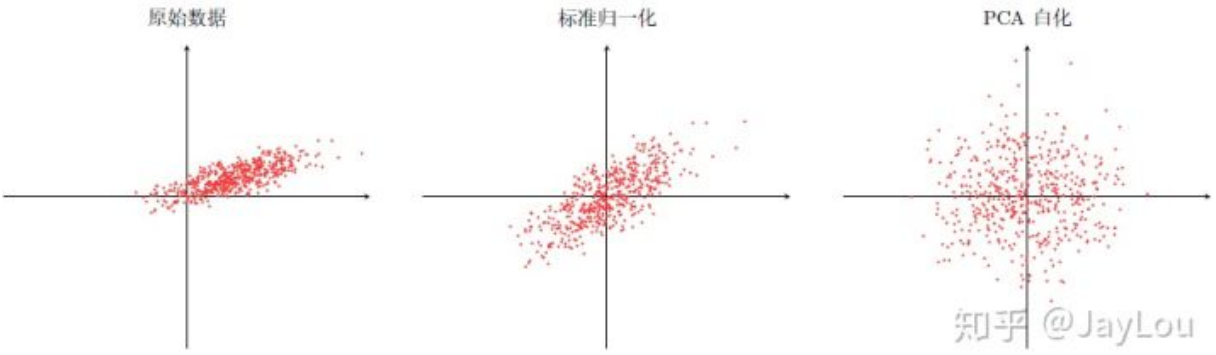
1、神经网络优化的难点体现在哪里？

深层神经网络是一个高度非线性的模型，其风险函数是一个非凸函数，因此风险最小化是一个非凸优化问题，会存在很多局部最优点。在高维空间中，非凸优化的难点并不在于如何逃离局部最优点，而是如何**逃离鞍点**。

2、神经网络数据预处理方法有哪些？神经网络怎样进行参数初始化？参数初始化为0、过大、过小会怎样？

（1）神经网络数据预处理方法有哪些？

缩放归一化：通过缩放将每一个特征的取值范围归一到[0, 1] 或[−1, 1] 之间
标准归一化：将每一个维特征都处理为符合标准正态分布（均值为0，标准差为1）。
白化（Whitening）：是一种重要的预处理方法，用来降低输入数据特征之间的冗余性。输入数据经过白化处理后，特征之间相关性较低，并且所有特征具有相同的方差。



（2）参数初始化为0、过大、过小会怎样？

参数为0：在第一遍前向计算时，所有的隐层神经元的激活值都相同。这样会导致深层神经元没有区分性。这种现象也称为**对称权重**现象。
参数过大或过小：参数过小还会使得sigmoid型激活函数丢失非线性的能力，这样多层神经网络的优势也就不存在了。如果参数取得太大，会导致输入状态过大。对于sigmoid 型激活函数来说，激活

值变得饱和，从而导致梯度接近于0。

（3）经常使用的初始化方法有以下几种：

Gaussian 分布初始化：

$$\mathcal{N}(0, \sqrt{\frac{2}{n_{in} + n_{out}}})$$

Xavier均匀分布初始化：参数可以在[−r, r] 内采用均匀分布进行初始化

对于sigmoid 型激活函数：

$$r = \sqrt{\frac{6}{n^{l-1} + n^{l}}},$$

对于tanh函数：

$$r = 4\sqrt{\frac{6}{n^{l-1} + n^{l}}}.$$

3、神经网络优化方法有哪些？

几种优化方法大体上可以分为两类：一是调整学习率，使得优化更稳定；二是调整梯度方向，优化训练速度。如图所示：

学习率衰减	梯度方向优化
AdaGrad、RMSprop、AdaDelta	动量法、Nesterov 加速梯度、梯度截断
Adam≈动量法+RMSprop	

AdaGrad：Adagrad 算法的缺点是在经过一定次数的迭代依然没有找到最优点时，由于这时的学习率已经非常小，很难再继续找到最优点。在第t 迭代时，计算每个参数梯度平方的累计值：

$$G_t = \sum_{\tau=1}^{t} \mathbf{g}_\tau \odot \mathbf{g}_\tau,$$

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t,$$

RMSprop：计算每次迭代梯度gt 平方的指数衰减移动平均：

$$\begin{aligned} G_t &= \beta G_{t-1} + (1-\beta)\mathbf{g}_t \odot \mathbf{g}_t \\ &= (1-\beta)\sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{g}_\tau \odot \mathbf{g}_\tau, \end{aligned}$$

动量法：用梯度的移动平均来代替每次的实际梯度：
Adam：Adam 算法一方面计算梯度平方的指数加权平均（和RMSprop 类似），另一方面计算梯度gt 的指数加权平均（和动量法类似）

$$\begin{aligned} M_t &= \beta_1 M_{t-1} + (1-\beta_1)\mathbf{g}_t, \\ G_t &= \beta_2 G_{t-1} + (1-\beta_2)\mathbf{g}_t \odot \mathbf{g}_t, \end{aligned}$$

$$\begin{aligned} \hat{M}_t &= \frac{M_t}{1-\beta_1^t}, \\ \hat{G}_t &= \frac{G_t}{1-\beta_2^t}. \end{aligned}$$

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{\hat{G}_t + \epsilon}}\hat{M}_t,$$

4、请介绍逐层归一化（Batch Normalization和Layer Normalization）？

（1）为什么要进行逐层归一化？什么是内部协变量偏移？

> 在深层神经网络中，中间某一层的输入是其之前的神经层的输出。因此，其之前的神经层的参数变化会导致其输入的分布发生较大的差异。在使用随机梯度下降来训练网络时，每次参数更新都会导致网络中间每一层的输入的分布发生改变。越深的层，其输入的分布会改变得越明显。**就像一栋高楼，低楼层发生一个较小的偏移，都会导致高楼层较大的偏移。**

协变量偏移：协变量是一个统计学概念，是可能影响预测结果的统计变量。 在机器学习中，协变量可以看作是输入。一般的机器学习算法都要求输 入在训练集和测试集上的分布是相似的。如果不满足这个要求，这些学 习算法在测试集的表现会比较差。

从机器学习角度来看，如果某个神经层的输入分布发生了改变，那么其参数需要重新学习，这种现象叫做**内部协变量偏移。**

内部协变量偏移会导致什么问题？

简而言之，每个神经元的输入数据不再是“独立同分布”。

上层参数需要不断适应新的输入数据分布，降低学习速度。
下层输入的变化可能趋向于变大或者变小，导致上层落入饱和区，使得学习过早停止。
每层的更新都会影响到其它层，因此每层的参数更新策略需要尽可能的谨慎。

为了解决内部协变量偏移问题，就要使得每一个神经层的输入的分布在训练过程中要保持一致。最简单直接的方法就是对每一个神经层都进行归一化操作，使其分布保存稳定。下面介绍几种比较常用的逐层归一化方法：**批量归一化、层归一化。**层归一化和批量归一化整体上是十分类似的，差别在于归一化的方法不同。

动机

训练的本质是学习数据分布。如果训练数据与测试数据的分布不同会**降低**模型的**泛化能力。**因此，应该在开始训练前对所有输入数据做归一化处理。
而在神经网络中，因为**每个隐层**的参数不同，会使下一层的输入发生变化，从而导致每一批数据的分布也发生改变；**致使**网络在每次迭代中都需要拟合不同的数据分布，增大了网络的训练难度与**过拟合**的风险。

（2）批量归一化（Batch Normalization，BN）的主要作用是什么？主要原理是什么？

BN 是一种**正则化**方法（减少泛化误差），主要作用有：

加速网络的训练（缓解梯度消失，支持更大的学习率）

防止过拟合：BN 可以看作在各层之间加入了一个新的计算层，**对数据分布进行额外的约束**，从而增强模型的泛化能力；
降低了**参数初始化**的要求。

基本原理

BN 方法会针对**每一批数据**，在**网络的每一层输入**之前增加**归一化**处理，使输入的均值为 `0` ，标准差为 `1` 。**目的**是将数据限制在统一的分布下。
具体来说，针对每层的第 `k` 个神经元，计算**这一批数据**在第 `k` 个神经元的均值与标准差，然后将归一化后的值作为该神经元的激活值。

$$\hat{x}_k \leftarrow \frac{x_k - \mathrm{E}[x_k]}{\sqrt{\mathrm{Var}[x_k]}}$$

BN 可以看作在各层之间加入了一个新的计算层，**对数据分布进行额外的约束**，从而增强模型的泛化能力；
但同时 BN 也降低了模型的拟合能力，破坏了之前学到的**特征分布**；为了**恢复数据的原始分布**，BN 引入了一个**重构变换**来还原最优的输入数据分布

$$y_k \leftarrow \gamma \hat{x}_k + \beta$$

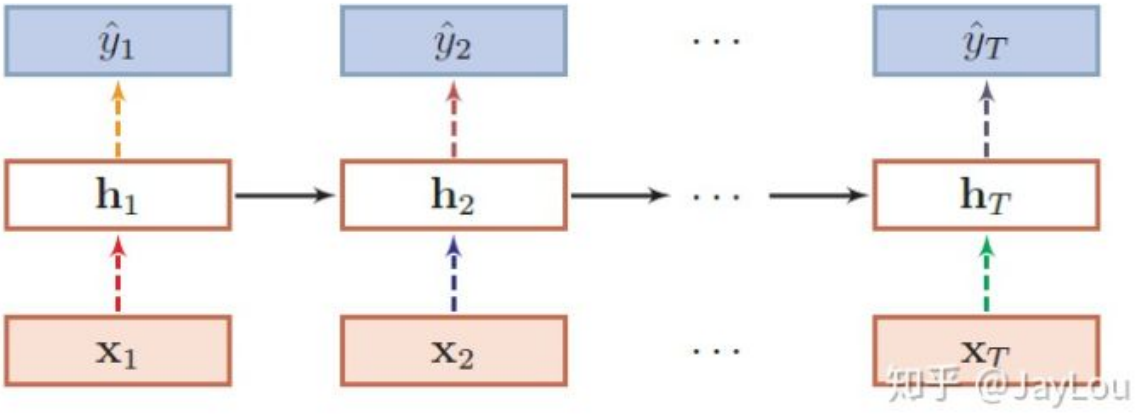
其中 `γ` 和 `β` 为可训练参数。

（3）BN 在训练和测试时分别是怎么做的？

训练时每次会传入一批数据，做法如前述；训练时**不采用移动平均**，使用 BN 的目的就是为了保证每批数据的分布稳定，使用全局统计量反而违背了这个初衷；
当测试或预测时，每次可能只会传入单个数据，此时模型会使用全局统计量代替批统计量（移动平均（moving averages））

5、神经网络正则化的方法有哪些？

L1 和L2 正则化：L1 和L2 正则化是机器学习中最常用的正则化方法，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
Batch Normalization（同上）
提前停止: 当验证集上的错误率不再下降，就停止迭代。
Dropout：集成学习的解释:每做一次丢弃，相当于从原始的网络中采样得到一个子网络。每次迭代都相当于训练一个不同的子网络，这些子网络都共享原始网络的参数。那么，最终的网络可以近似看作是集成了指数级个不同网络的组合模型。**当在循环神经网络上应用丢弃法**，不能直接对每个时刻的隐状态进行随机丢弃，这样会损害循环网络在时间维度上记忆能力。一种简单的方法是对非时间维度的连接（即非循环连接）进行随机丢失:



虚线边表示进行随机丢弃，不同的颜色表示不同的丢弃掩码

数据增强： 增加数据量，提高模型鲁棒性，避免过拟合。目前，数据增强还主要应用在图像数据上，在文本等其它类型的数据还没有太好的方法。
标签平滑：在输出标签中添加噪声来避免模型过拟合。

6、神经网络怎么解决梯度消失问题？

选择合适的激活函数：前馈神经网络：ReLU 循环神经网络：tanh
Batch Normalization
采取残差网络ResNet

说明：本文回答及说明来自《神经网络与深度学习》