1、 在前馈神经网络中，所有的参数能否被初始化为 0？如果不能，参数能否全部被初始化为其他相同的值？并给出具体理由。

答：

前置知识：神经网络结构可以分为前馈网络和循环网络。前馈神经网络（Feedforward Neural Network，FNN）是最基本的一种人工神经网络结构，它由多层节点组成，每层节点之间是全连接的，即每个节点都与下一层的所有节点相连。前馈神经网络的特点是信息只能单向流动，即从输入层到隐藏层，再到输出层，不能反向流动。它可以通过反向传播算法来调整权值以进行训练。

在前馈神经网络中，所有的参数通常不能被初始化为 0。原因如下：

1.对称性破缺：如果所有权重初始化为 0，那么网络中的所有神经元在前向传播时将产生相同的输出，这会导致网络无法学习到数据的复杂特征。同样，如果所有权重初始化为相同的非零值，也会导致类似的对称性问题，因为每个神经元将对输入数据做出相同的响应。

2.梯度消失或爆炸：在反向传播过程中，如果所有权重相同，那么梯度更新将对所有权重产生相同的影响，这可能导致梯度消失或爆炸问题，使得网络难以训练。

3.独立性：每个神经元应该能够独立地学习不同的特征，如果权重相同，这种独立性就无法实现。

也不能初始化为相同的值，理由与上述相同。

我们初始化权值为

$$W1=\begin{bmatrix} w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad W2=\begin{bmatrix} w_{64} & w_{65} \end{bmatrix}=\begin{bmatrix} 0 & 0 \end{bmatrix}$$

其中W1代表**输入层到隐藏层**的权值矩阵，W2代表**隐藏层到输出层**的权值矩阵。

假设网络的输入为[x1,x2,x3]，然后通过网络的正向传播，可以得出：

$$z_4 = w_{41} * x_1 + w_{42} * x_2 + w_{43} * x_3$$
$$z_5 = w_{51} * x_1 + w_{52} * x_2 + w_{53} * x_3$$

由于

$$W1=\begin{bmatrix} w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

我们可以知道:

$$z_4 = z_5 = 0$$

从上面可以知道，此时隐藏层的值是相同的，然后经过激活函数f后，**得到的a4,a5仍然是相同的**，如下：

$$a_4 = a_5 = f(z_4)$$

最终网络的输出为：

$$a_6 = w_{64} * a_4 + w_{65} * a_5 = 0$$

此时，假设我们的真实输出为y,则均方误差损失函数可以表示为：

$$Loss = \frac{1}{2}(y-a_6)^2$$

到了这里，此时又应该到我们伟大的BP反向传播算法出场了！**我们需要反向更新权值，它使得预测的输出值与真实值越来越靠近。**

这里假设我们的读者已经知道了BP反向传播的过程

可以知道，通过反向传播后，结点4,5的梯度改变是一样的，假设都是 $\triangle w$,那么此时结点4与结点6之间的参数，与结点5与结点6之间的参数变为了，如下：

$$w_{64} = 0 + \Delta w = \Delta w$$
$$w_{65} = 0 + \Delta w = \Delta w$$

由上式可以看出，新的参数相同了！！！！

同理可以得出输入层与隐藏层之间的参数更新都是一样的，得出更新之后的参数

$$w_{41}, w_{42}, w_{43}, w_{51}, w_{52}, w_{53}$$

都是相同的！然后不管进行多少轮正向传播以及反向传播，每俩层之间的参数都是一样的。

换句话说，本来我们希望不同的结点学习到不同的参数，但是由于参数相同以及输出值都一样，不同的结点根本无法学到不同的特征！这样就失去了网络学习特征的意义了。

隐藏层与其它层多个结点，其实仅仅相当于一个结点！！如下图表示：

通常采用随机初始化的方法来设置权重，这样可以打破对称性，允许每个神经元学习不同的特征。偏置通常初始化为较小的正值，以确保在训练开始时神经元的输出不会全部为 0。常见的初始化方法包括：

1.Xavier 初始化：根据前一层的神经元数量来设置权重的初始分布，适用于 sigmoid 和 tanh 激活函数。

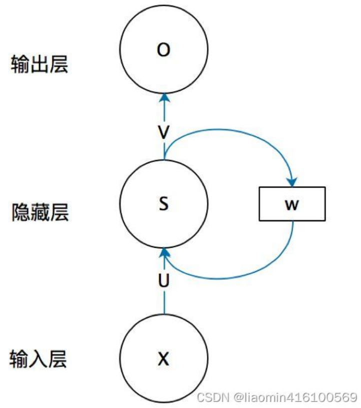
2.He 初始化：类似于 Xavier 初始化，但适用于 ReLU 激活函数。

3.随机正态分布：权重从具有特定均值和方差的正态分布中随机抽取。

2、 请自行学习理解长短时记忆网络（LSTM），思考 LSTM 是否能解决神经网络中遇到的梯度消失和梯度爆炸问题，并给出详细的说明。

答：

循环神经网络（RNN）是一种用于处理序列数据的神经网络架构。它与传统的前馈神经网络不同，RNN 具有循环连接，允许信息在网络中随时间步长传递。这种结构使得 RNN 特别适合于处理时间序列数据、自然语言处理、语音识别等领域的问题。



现在看上去就会清楚许多，这个网络在t时刻接收到输入x_t之后，隐藏层的值是s_t，输出值是o_t。关键一点是s_t的值不仅仅取决于x_t，还取决于s_{t-1}。

公式1：$s_t = f(U * x_t + W * s_{t-1} + B1)$

公式2：$o_t = g(V * s_t + B2)$

- 式1是隐藏层的计算公式，它是循环层。U是输入x的权重矩阵，W是上一次隐藏层值S_{t-1}作为这一次的输入的权重矩阵，f是激活函数。
- 式2是输出层的计算公式，V是输出层的权重矩阵，g是激活函数,B1,B2是偏置假设为0。

x 是输入向量，o 是输出向量，s 表示隐藏层的值；U 是输入层到隐藏层的权重矩阵，V 是隐藏层到输出层的权重矩阵。循环神经网络的隐藏层的值 s 不仅仅取决于当前这次的输入 x，还取决于上一次隐藏层的值 s。权重矩阵 W 就是隐藏层上一次的值作为这一次的输入的权重。

RNN 的关键特点包括：

共享权重、循环连接、隐藏状态、梯度消失和梯度爆炸。

梯度消失与梯度爆炸：

梯度消失是指在神经网络的反向传播过程中，由于连续乘积的链式法则，梯度值随着层数的增加而逐渐减小，最终变得非常接近于零。这导致网络中靠近输入层的权重更新非常缓慢，甚至几乎不更新，使得训练过程变得非常低效，甚至无法进行。

梯度爆炸是指在反向传播过程中，梯度值随着层数的增加而迅速增大，最终变得非常大，导致权重更新过大，使得网络训练变得不稳定，甚至发散。

假设我们的时间序列只有三段，S_0为给定值，神经元没有激活函数，则RNN最简单的前向传播过程如下：

$$S_1 = W_x X_1 + W_s S_0 + b_1$$
$$O_1 = W_o S_1 + b_2$$
$$S_2 = W_x X_2 + W_s S_1 + b_1$$
$$O_2 = W_o S_2 + b_2$$
$$S_3 = W_x X_3 + W_s S_2 + b_1$$
$$O_3 = W_o S_3 + b_2$$

输入时间序列长度为t的数据，假设在t时刻，损失函数为$L_t = \frac{1}{2}(Y_t - O_t)^2$。

使用随机梯度下降算法训练RNN，其实就是对W_x、W_s、W_o以及b_1、b_2求偏导，并不断调整它们，使得L_t尽可能小的过程。

现在假设我们的时间序列只有3段，t_1、t_2、t_3。

我们对t_3时刻的W_x、W_s、W_o求偏导（其他时刻类似）：

$$\frac{\partial L_3}{\partial W_0} = \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial W_o}$$

$$\frac{\partial L_3}{\partial W_x} = \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial W_x} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial W_x} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial S_1}\frac{\partial S_1}{\partial W_x}$$

$$\frac{\partial L_3}{\partial W_s} = \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial W_s} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial W_s} + \frac{\partial L_3}{\partial O_3}\frac{\partial O_3}{\partial S_3}\frac{\partial S_3}{\partial S_2}\frac{\partial S_2}{\partial S_1}\frac{\partial S_1}{\partial W_s}$$

可以看出对于W_0求偏导并没有长期依赖，但是对于W_x、W_s求偏导，会随着时间序列产生长期依赖。因为S_t随着时间序列向前传播，而S_t又是W_x、W_s的函数。

根据上述求偏导的过程，我们可以得出任意时刻对W_x、W_s求偏导的公式：

$$\frac{\partial L_t}{\partial W_x} = \sum_{k=0}^{t}\frac{\partial L_t}{\partial O_t}\frac{\partial O_t}{\partial S_t}\left(\prod_{j=k+1}^{t}\frac{\partial S_j}{\partial S_{j-1}}\right)\frac{\partial S_k}{\partial W_x}$$

$$\frac{\partial L_t}{\partial W_s} = \sum_{k=0}^{t}\frac{\partial L_t}{\partial O_t}\frac{\partial O_t}{\partial S_t}\left(\prod_{j=k+1}^{t}\frac{\partial S_j}{\partial S_{j-1}}\right)\frac{\partial S_k}{\partial W_s}$$

如果加上激活函数，$S_j = tanh(W_x X_j + W_s S_{j-1} + b_1)$，

则$\prod_{j=k+1}^{t}\frac{\partial S_j}{\partial S_{j-1}} = \prod_{j=k+1}^{t} tanh^{'} W_s$

由于激活函数tanh的导数是小于1的，因此随着累乘的增加，RNN会出现梯度消失的情况。

如果梯度的值非常大，那么靠近输入层的梯度将迅速增长，导致梯度爆炸；如果梯度的值非常小，那么靠近输入层的梯度将迅速减小，导致梯度消失（有些常用的激活函数（如 Sigmoid 和 Tanh）在输入值较大或较小时，它们的导数趋近于零，容易导致梯度消失）。

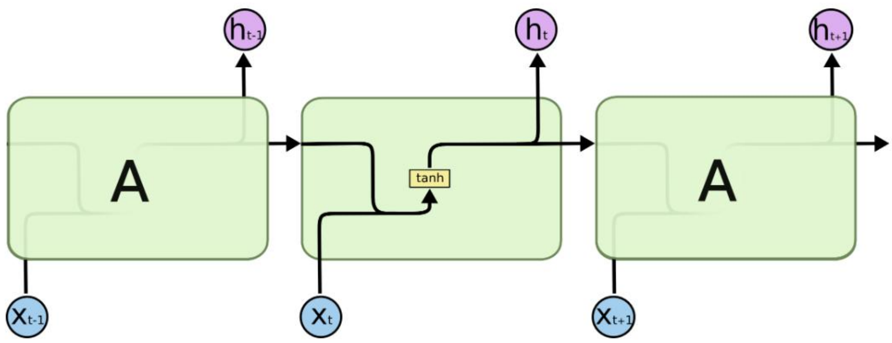
长短期记忆网络（LSTM）是一种特殊类型的循环神经网络（RNN）。**LSTM 旨在解决传统 RNN 在处理长序列数据时遇到的梯度消失或梯度爆炸问题。**原始 RNN 的隐藏层只有一个状态，即 h，它对于短期的输入非常敏感。那么如果我们再增加一个门（gate）机制用于控制特征的流通和损失，即 c，让它来保存长期的状态，这就是长短时记忆网络。

LSTM 网络的核心是三个门的机制：遗忘门（forget gate）、输入门（input gate）、输出门（output gate）。这些门通过自适应的方式控制信息的流动，从而实现对长期依赖信息的捕捉。

原理：

RNN：

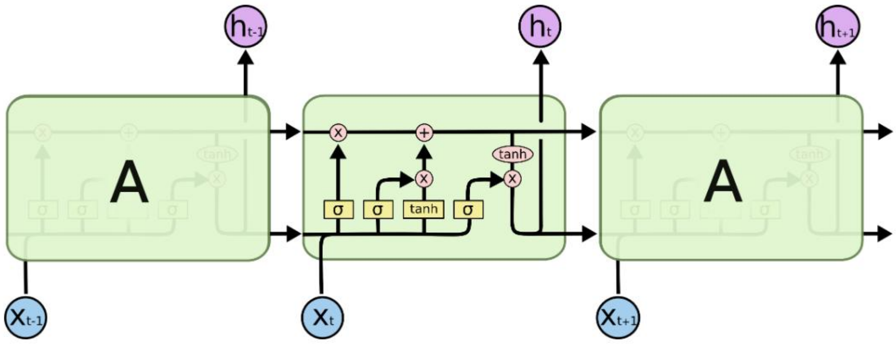
在标准的 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层。



LSTM:

LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。具体来说，RNN 是重复单一的神经网络层，LSTM 中的重复模块则包含四个交互的层，三个 Sigmoid 和一个 tanh 层，并以一种非常特殊的方式进行交互。图中σ表示的 Sigmoid 激活函数与 tanh 函数类似，不同之处在于 sigmoid 是把值压缩到 0~1 之间而不是 -1~1 之间。这样的设置有助于更新或

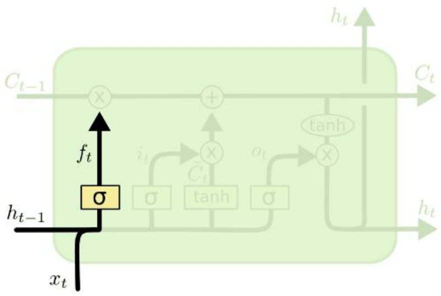
忘记信息，相当于要么是 1 则记住，要么是 0 则忘掉。



LSTM 的关键就是细胞状态，水平线在图上方贯穿运行。细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。LSTM 拥有三种类型的门结构：遗忘门/忘记门、输入门和输出门，来保护和控制细胞状态。

遗忘门：

在我们 LSTM 中的第一步是决定我们会从细胞状态中丢弃什么信息。这个决定通过一个称为“忘记门”的结构完成。该忘记门会读取上一个输出和当前输入，做一个 Sigmoid 的非线性映射，然后输出一个向量（该向量每一个维度的值都在 0 到 1 之间，1 表示完全保留，0 表示完全舍弃，相当于记住了重要的，忘记了无关紧要的），最后与细胞状态相乘。



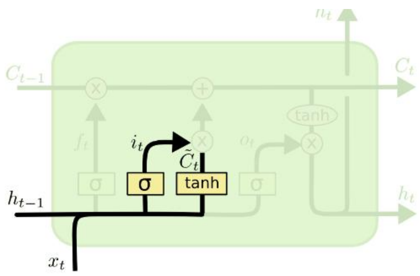
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

输入门：

下一步是确定什么样的新信息被存放在细胞状态中。这里包含两个部分：

第一，sigmoid 层称“输入门层”决定什么值我们将要更新；

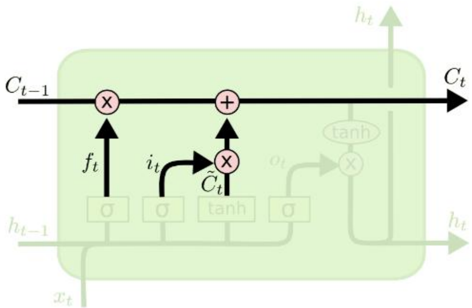
第二，一个 tanh 层创建一个新的候选值向量，会被加入到状态中。



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

更新细胞状态：

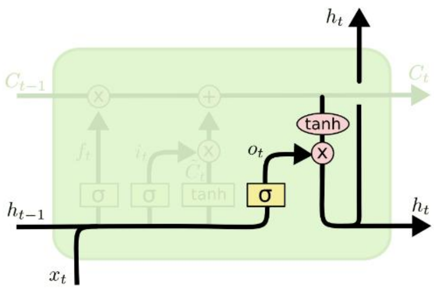
我们把旧状态与 ft 相乘，丢弃掉我们确定需要丢弃的信息，接着加上 it*Ct。这就是新的候选值，根据我们决定更新每个状态的程度进行变化。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

输出门:

最终，我们需要确定输出什么值。这个输出将会基于我们的细胞状态，但是也是一个过滤后的版本。首先，我们运行一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。接着，我们把细胞状态通过 tanh 进行处理（得到一个在-1 到 1 之间的值）并将它和 sigmoid 门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

因此，LSTM 确实能够有效解决神经网络中常见的梯度消失和梯度爆炸问题。它通过引入记忆单元和三个门控机制（输入门、遗忘门和输出门）来有选择性地保留或更新长期重要的参数。这种设计使得 LSTM 能够控制信息的流动，从而缓解梯度消失和梯度爆炸的问题。LSTM 最初就是为了解决这些问题而被设计的。