Q1：常见的激活函数有哪些，它们的特点以及适用的场景？

常见的激活函数：

(1)Sigmoid 函数

$$f(x)=\frac{1}{1+\exp(-x)} \tag{1}$$

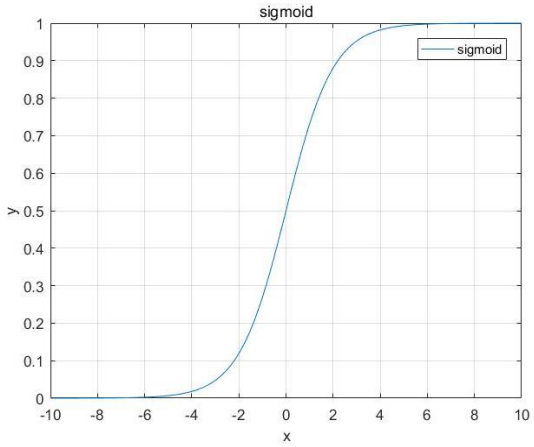


图 1 Sigmoid 函数曲线图

特点：经过 Sigmoid 函数后，输出响应的值域被压缩到(0,1)。求导时中间梯度大，两端梯度小，因此会导致该区域的导数在深度网络的反向误差传播过程中出现梯度消失现象。

（Logistic Regression 选取 Sigmoid 作为激活函数难道仅因为其形状可以满足对条件概率的要求嘛？查阅资料的时候大多数提到的都是这一点。这里可以帮我解释一下不？）

(2)tanh(x)函数

$$\mathrm{Tanh}(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

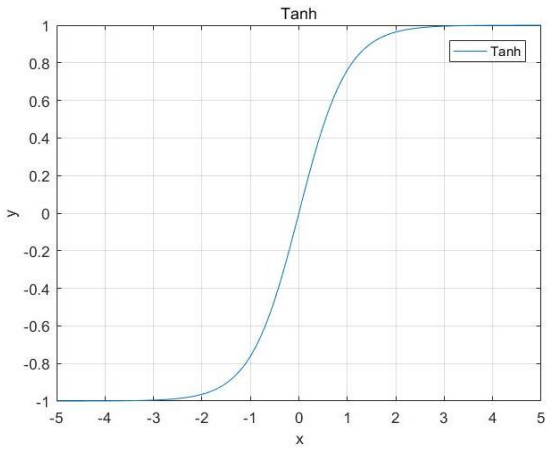


图 2 tanh 函数曲线图

特点：将输出响应映射到(-1,1)之间，且关于(0,0)中心对称，但和 Sigmoid 函数一样仍然存在梯度消失现象。

(3)ReLU 函数

$$\text{ReLU\_g}(x)=\begin{cases} x & ,(x>0) \\ 0 & ,(x\le 0) \end{cases} \tag{3}$$

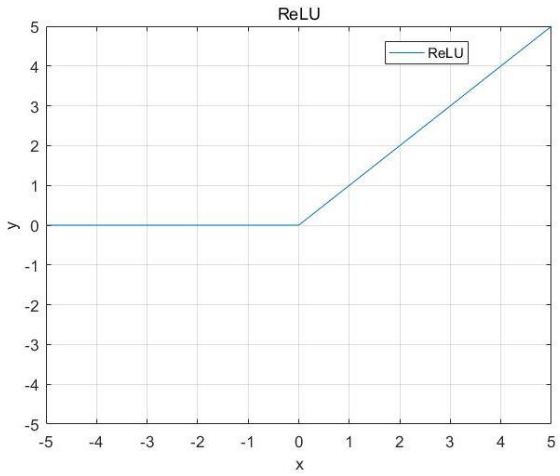


图 3 ReLU 函数曲线图

特点：函数在输入大于 0 时，直接输出该值；在输入小于 0 时，输出 0。消除了 Sigmoid 和 Tanh 函数存在的梯度消失现象，且在求导过程中计算简单。因其是分段线性函数即单侧抑制，使得神经网络中的神经元有稀疏激活性，模型能够更好地挖掘相关特征，拟合训练数据。

Q2：正则化的本质是什么以及常用的正则化方法有哪些？

训练神经网络的目的是使得训练得到的参数是损失函数最小的那一组，但是使得损失函数最小所对应的参数不唯一，可大可小，倘若所对应的参数很大的话，那么在一个新样本上进行预测，由于参数(w,b)很大使得噪声也被放大，进而影响判断结果，因此需要在损失函数后加入正则项。

加入正则项可以削弱特征之间的耦合关系，即特征稀疏化；

从权重衰减的角度考虑：

没有加正则项的损失函数可以写成 $J(w,b)$，权重更新过程表示为：

$$w=w-\eta\nabla_w J(w) \tag{4}$$

加入正则项后的损失函数可以写为：

$$\hat{J}(w)=J(w)+\lambda w^T w \tag{5}$$

为了方便后续求导过程将 λ 替换为 $\frac{\alpha}{2}$：

$$\hat{J}(w)=J(w)+\frac{\alpha}{2}w^T w \tag{6}$$

权重更新过程如下：

$$
\begin{aligned}
w &= w - \eta \nabla_w \hat{J}(w) \\
&= w - \eta [\nabla_w J(w) + \alpha w] \\
&= w - \eta \nabla_w J(w) - \eta \alpha w \\
&= (1 - \eta \alpha) w - \eta \nabla_w J(w)
\end{aligned} \tag{7}
$$

对比式(4)和(7)可以从看出 w 的系数看出，正则化使得权重得以衰减。调整权重 w，使得 $w \to 0$，那么神经网络拟合出的函数的高次项也得以抑制。进而减少过拟合。

常用正则化的方法：

L1、L2 正则、数据增强、Early Stopping、Dropout

Q3：梯度是什么，发生梯度消失、梯度爆炸可能的原因有哪些？

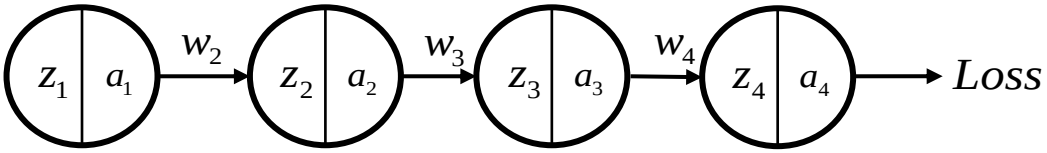


图 4 参数传递过程示意图

其中 $a_j = \sigma(z_j)$，$\sigma(\cdot)$ 为激活函数，$z_j = w_j \cdot a_{j-1} + b_j$ 为神经元的输入。

损失函数关于第一个神经元的梯度为：

$$
\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial a_4} \times \sigma'(z_4) \times w_4 \times \sigma'(z_3) \times w_3 \times \sigma'(z_2) \times w_2 \times \sigma'(z_1) \tag{8}
$$

当偏置 b_1 发生变化时，第一个神经元的输出 a_1 也很会受影响：

$$
\begin{aligned}
\Delta a_1 &= \frac{\partial \sigma(w_1 a_0 + b_1)}{\partial b_1} \times \Delta b_1 \\
&= \sigma'(z_1) \Delta b_1
\end{aligned} \tag{9}
$$

a_1 受影响后后面的神经元的输出都会改变。

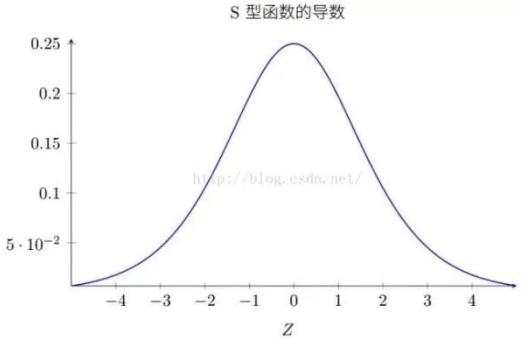


图 5 Sigmoid 导数图

如果激活函数选择 Sigmoid 函数（其导数如图 5 所示），在输入为 0 时，导数值最大，且最大导数值小于 1，从(8)式中可以看出，连乘小于 1 的导数越多，导致梯度下降的越快，传播过程中就会出现梯度消失现象。

梯度爆炸现象：

假设网络的权重很大，再通过调整神经元偏置使得$\sigma'(z_j)$保持在最大值范围内，这样通过(8)式的传播机制，梯度反传过程中就会出现梯度爆炸现象。

出现梯度消失、梯度爆炸的原因：
网络过深、选择 Sigmoid 作为神经元的激活函数。

Q4：梯度下降的原理是什么？有哪几种梯度下降的方式？

梯度下降法是最为常用的最优化方法。当目标函数是凸函数时，梯度下降法的解是全局解。梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以也被称为是”最速下降法“。最速下降法越接近目标值，步长越小，前进越慢，步长越大，误差越大。

梯度下降法表达式：

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \tag{10}$$

其中 $J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$ 为损失函数，α 为学习率，通过式(10)同步更新 θ_0 和 θ_1 直到使得 $J(\theta_0,\theta_1)$ 最小。

梯度下降方式：

(1)批量梯度下降法

假设要拟合的函数为：

$$h(\theta)=\sum_{j=0}^{n}\theta_j x_j \tag{11}$$

损失函数为：

$$J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2 \tag{12}$$

将损失函数对参数求导：

$$\frac{\partial J(\theta)}{\partial \theta_j}=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{i})-y^{i})x_j^i \tag{13}$$

对参数进行更新：

$$\theta_j'=\theta_j-\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{i})-y^{i})x_j^i \tag{14}$$

一次迭代对所有样本进行计算，但是当样本数 m 很大时，训练速度很慢。

(2)随机梯度下降法

每次更新参数不需要再全部遍历一次整个样本，只需要查看一个训练样本进行更新，之后再用下一个样本进行下一次更新。虽然不是每次迭代得到的损失函数都向着全局最优方向，但是大的整体的方向是向全局最优解的，适用于大规模训练样本情况。

Q5：模型反向传播的原理是什么？

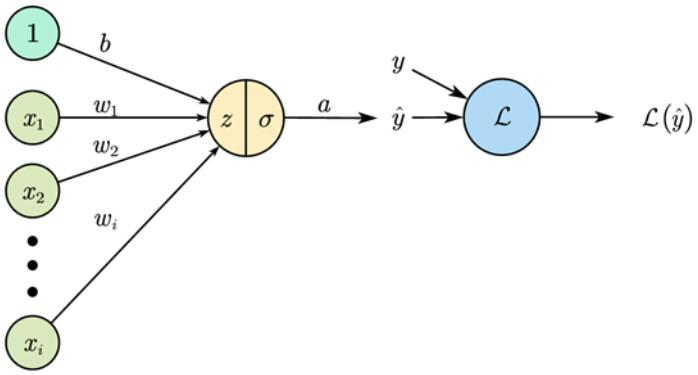


图 6 单个神经元结构图

输入为 $x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix}^T$，权重为 $w=\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_i \end{bmatrix}$，偏置为 b，激活函数为 $\sigma(\cdot)$，输出为 $\hat{y}$，则，损失函数为 $L(\hat{y})$：

$$z=\sum w_i x_i + b \tag{15}$$

$$a=\sigma(z) \tag{16}$$

损失函数对权重求偏导：

$$\frac{\partial L(\hat{y})}{\partial w}=\frac{\partial L(\hat{y})}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w} \tag{17}$$

$$\frac{\partial L(\hat{y})}{\partial b}=\frac{\partial L(\hat{y})}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial b} \tag{18}$$

参数更新：

$$w \leftarrow w-\eta\Delta w \tag{19}$$

$$b \leftarrow b-\eta\Delta b \tag{20}$$

反向传播主要是基于链式法则对参数逐层进行求导。

Q6：归一化方法有哪些？为什么要对数据进行归一化处理？

(1)为什么要对数据进行归一化？

①因为有些特征的数值比其他特征的数值高，归一化可以缓和差异，在梯度下降求解最优解时能较快地收敛；

②在训练过程中由于网络参数的变化导致网络激活分布变化，通过归一化操作减少内部协方差的变化，从而提高训练效果。

(2)归一化方法

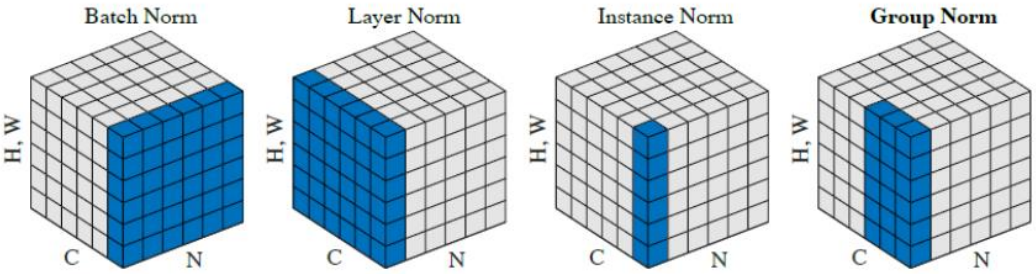


图 7 四种常见归一化方法示意图

①Batch Normalization

批量归一化通过在训练阶段稳定层输入的分布，提高训练效果。这种方法主要和内部协方差有关。

内部协方差指的是由于每一层参数更新使得上一层数据布改变，并且随着网络层数的加深而变得更加严重，这就使得高层需要不断地去重新适应底层的参数更新，由于底层的参数随着训练更新，导致相同输入分布得到的输出分布改变了。因此我们可以通过批量归一化操作，控制层数据的均值和方差来减少内部协方差偏移现象。

②Layer Normalization

批量归一化操作是针对同一层单个神经元的，同层不同的神经元经过再次平移和缩放后再分布到不同的区间中，而层归一化是对同一层的神经元做归一化处理：

$$Batch\_Normalization:$$
$$\mu^{(l)} = \frac{1}{K}\sum_{k=1}^{K} z^{(k,l)} \tag{21}$$

批量归一化求第 l 层输入 $z^{(l)}$ 的每一维的期望和方差，K 为批量里的样本数量。

$$Layer\_Normalization:$$
$$\mu^{(l)} = \frac{1}{n^{(l)}}\sum_{i=1}^{n^{(l)}} z_i^{(l)} \tag{22}$$

其中 $n^{(l)}$ 为第 l 层神经元的数量。比如 Batch_Size 为 8，会有 8 个均值和方差

被同时求出，每个均值和方差都是由单个图片的所有通道之间做归一化操作得到的。

③Instance Normalization

实例归一化是对一个批次中的单个图片进行归一化而不是像批量归一化对整个批次中的所有图片进行归一化。

④Group Normalization

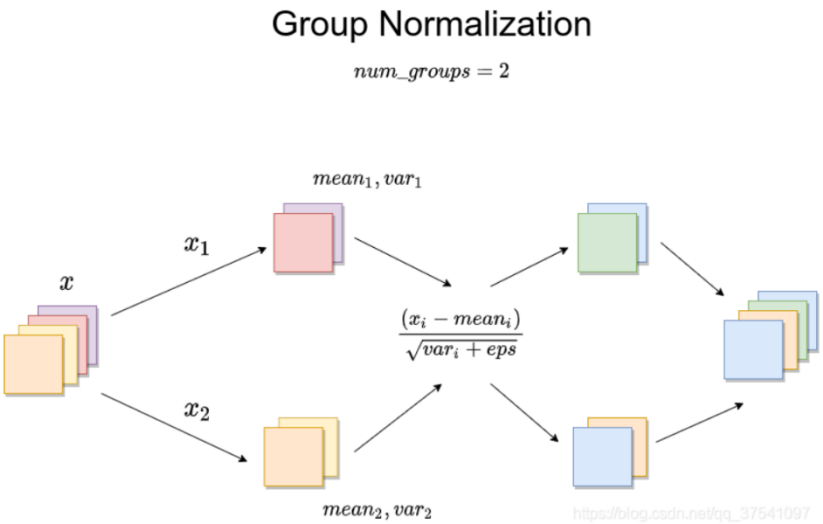


图 8 分组归一化方法示意图

分组归一化操作如图 8 所示，假设 num_groups=2，某层输出得到 x，根据 num_groups 沿 channel 方向均分成 num_groups 份，然后对每一份求均值和方差。

Q7：池化方法有哪些？原理、区别、使用场景？

①平均池化

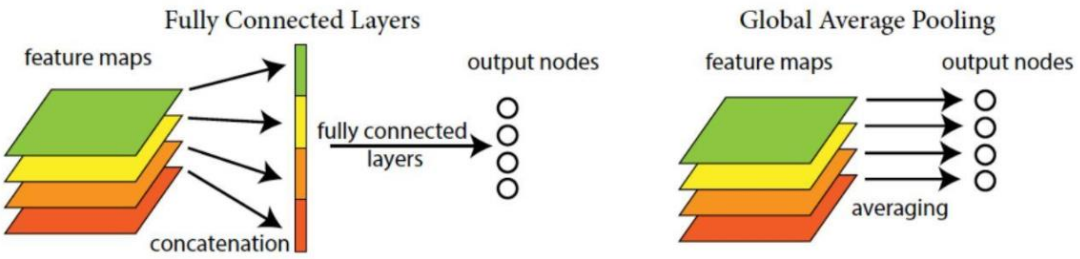
对领域内的点求平均，可以减小因领域大小受限造成的估计值方差增大，更多的保留背景信息（这里不太理解，可以帮忙解释一下不）。

②最大池化

对领域内的点取最大值。可以减小因卷积层参数误差造成的估计均值的偏移，更多地保留纹理信息（这里不太理解，可以帮忙解释一下不）。

③全局平均池化

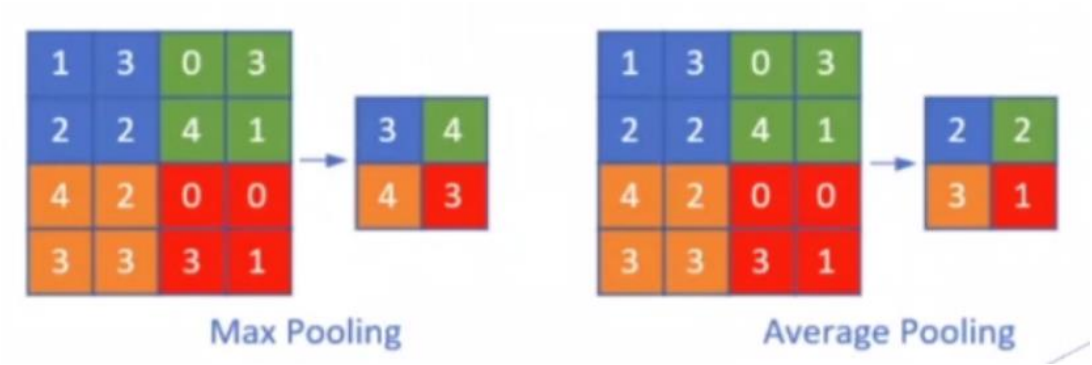
对于输出特征图的每一个通道的所有像素计算一个平均值，经过全局平均池化之后就得到一个维度等于类别数的特征向量，然后直接输入到 softmax 层，每个类别得到一个概率值，可以替代全连接层。



④随机池化

只需要对特征图中的元素按照其概率值大小随机选择，元素选中的概率与其数值大小正相关，并非如同最大池化那样直接选取最大值。随机池化操作不但最大化地保证了取值的 Max，也部分确保不会所有的元素都被选取 max 值，从而提高了泛化能力。

Q8：池化的前向过程和反向过程有什么区别？



最大/平均池化正向过程示意图

最大/平均池化反向过程示意图

在反向过程中都先还原为池化之前的大小。

对于最大池化，在前向过程中需要记录最大值的位置，在反向过程中将池化结果还原到对应的位置，其他位置均为 0；

对于平均池化的反向过程，对池化后的结果按原特征图的大小进行平均。