• A2.1 (5 points) 尝试解释Epoch、Iteration、Batch几个概念及其不同，尝试说明batch_size的选择依据和影响。

Epoch、Iteration、Batch几个概念及其不同：

Epoch指的是整个训练数据集被遍历一次的过程。也就是说，假设训练数据集有N个样本，那么在一次Epoch中，这N个样本都会被模型学习一次。通过多次Epoch的训练，模型可以逐步优化其参数，以更好地拟合数据。

Batch Size，即批大小，是指在每一次模型权重更新时所使用的样本数量。在深度学习中，我们通常不会一次性使用所有的训练数据来更新模型权重（虽然也有全局batch这种训法，但是还是很不推荐），而是将数据分成若干个批次，每个批次包含一定数量的样本。这样，模型可以在每个批次上进行权重更新，从而逐步优化其性能。

Iteration，即迭代次数，指的是模型在训练过程中完成一次Epoch所需的批次数量。换句话说，Iteration就是模型在整个训练数据集上完成一次遍历所需的步骤数。每完成一次Iteration，模型都会对所有批次的数据进行一次前向传播和一次反向传播，从而更新其权重。在实际应用中，我们通常需要设置足够的Iteration次数，以确保模型能够充分学习数据的内在规律和模式。

Batch Size和Iteration相乘会非常接近样本数N。至于为什么不是等于，因为有取整函数。

batch_size的选择依据和影响：

采用较小的Batch Size：

• 不容易过拟合，泛化能力较强。

• 梯度估计噪声较大，收敛速度较慢，训练时间会更长。

•计算占用内存较少。

采用较大的Batch Size：

• 容易过拟合，可能会导致模型陷入局部最小值。

• 梯度估计更准确，收敛速度较快，训练速度较快，但可能跳过最优解。

•计算占用内存较大。

在实际操作过程中，Batch Size的大小通常设置为2的n次幂，如32、64、128、256等，这可能是和CPU，GPU，NPU等的硬件构造有关。

32算是较小的Batch Size，64、128算中等的，256往上算比较大的。

如果说想快速收敛，推荐大Batch Size；想注重泛化能力，推荐小Batch Size，想减少计算所需内存，推荐小Batch Size。

不过实际的神经网络很复杂，最合适的Batch Size推荐靠实验来寻找。

• A2.2 (5 points) 以一个简单的1-1-1结构的两层神经网络为例，分别采用均方误差损失函数和交叉熵损失函数，说明这两种函数关于参数的非凸性（可作图示意和说明）。

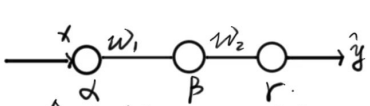
要证明一个函数是非凸的，最直接的方法是证明其Hessian矩阵不是半正定的。Hessian矩阵是由函数的二阶偏导数构成的矩阵。

我们知道，所有顺序主子式（即从左上角开始的各个子矩阵的行列式）都非负，那么A就是半正定的。所以，我们只要找到这个函数的Hessian矩阵的顺序主子式有一个符号不定（可以为负）的元素，那么就可以确定他是否具有半正定性。

该神经网络对于两个w参数的Hessian矩阵如下：

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial w_1^2} & \frac{\partial^2 L}{\partial w_1 \partial w_2} \\ \frac{\partial^2 L}{\partial w_2 \partial w_1} & \frac{\partial^2 L}{\partial w_2^2} \end{bmatrix}$$

我们只确定一个参数的二阶导的符号就可以了，推导如下：



令 $X = (x_1, x_2, \cdots x_n)$

$y = (y_1, y_2, \cdots y_n)$

若令 $z_1 = w_1 x + b_1$

$q_1 = f(z_1)$

$z_2 = w_2 q_1 + b_2$

其中，z_1 为 β 的输入，

q_1 为 β 的输出，

z_2 为 γ 的输入。

有 $\hat{y} = f(z_2)$

① 均方误差（假设 $f(x)$ 为 sigmod 函数）

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

$$\frac{dL}{dw_2} = \frac{dL}{dy}\frac{dy}{dz_2}\frac{dz_2}{dw_2} = (\hat{y} - y)(f(z_2))(1 - f(z_2))\, q_1$$

$$\frac{\partial^2 L}{\partial w_2^2} = 2[f(z_2)(1 - f(z_2))(1 - 2f(z_2)]\, f^2(z_1) + 2(\hat{y} - f(z_2))\, f''(z_2)\, f^2(z_1)$$

$$f'(z_2) = f(z_2)(1 - f(z_2))$$

$$f''(z_2) = f'(z_2)(1 - 2f(z_2))$$

② 交叉熵

$$L_{CE} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

$$\frac{\partial L_{CE}}{\partial w_2} = -\left[\frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}}\right] \cdot f'(z_2) f(z_1)$$

$$= -(y - f'(z_2)) \cdot f(z_1)$$

$$\frac{\partial^2 L_{CE}}{\partial w_2^2} = f(z_1)[1 - f(z_2)]\, f^2(z_1)$$

可确定符号

$$\frac{\partial^2 L(\theta)}{\partial w_1^2} = \frac{\partial}{\partial w_1}\left(\frac{\partial L(\theta)}{\partial w_1}\right) = \frac{\partial}{\partial w_1}\left[(f(z_2)-y)\cdot w_2 \cdot f(z_1)(1-f(z_1))\cdot x\right]$$

$$= \frac{\partial (f(z_2)-y)}{\partial w_1}\cdot w_2 \cdot f(z_1)(1-f(z_1))\cdot x + (f(z_2)-y)\cdot w_2 \frac{\partial}{\partial w_1}\left[f(z_1)(1-f(z_1))x\right]$$

$$= f(z_2)(1-f(z_2))\cdot w_2^2 \cdot f^2(z_1)(1-f(z_1))^2 \cdot x^2 + (f(z_2)-y)\cdot w_2 f'(z_1)(1-2f(z_1))\cdot x^2$$

符号不可确定，主要是后者的符号不定，可以变化的量太多，取绝对值存在为负且绝对值大于前者的情况

• A2.3 (5 points) 尝试推导：在回归问题中，假设输出中包含高斯噪音，则最小化均方误差等价于最小化负log似然。

假设我们有一个回归模型：y=f(x)+ϵ

其中：

- y 是输出变量。
- x 是输入特征向量。
- f(x) 是模型预测。
- ϵ 是噪声项，假设服从均值为 0、方差为 σ^2 的高斯分布，即 ϵ~N(0,σ²)。

给定一组数组集 $\{(x_i, y_i)\}_{i=1}^n$

在高斯噪声假设下，y_i 的概率密度函数为：

$$P(y_i|x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{(y_i - f(x_i))^2}{2\sigma^2}\right)}$$

由样本独立，训练集的似然函数为

$$L = \prod_{i=1}^n P(y_i|x_i) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\left(-\frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - f(x_i))^2\right)}$$

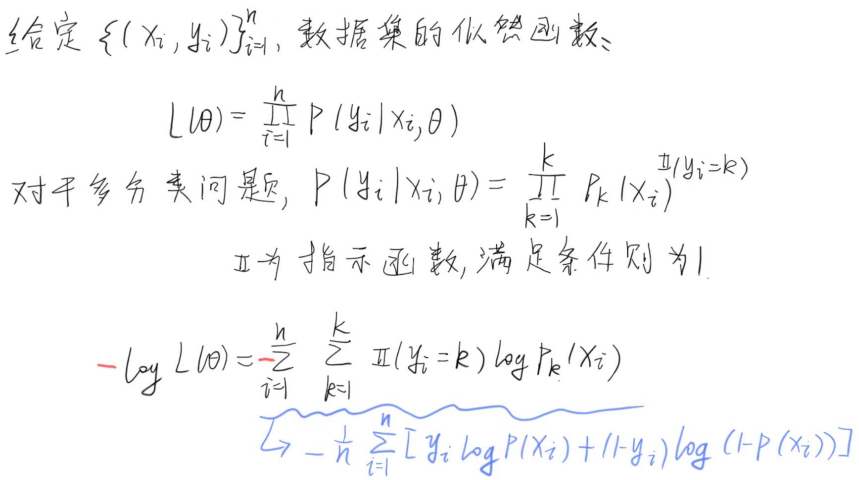
我们一般取对数：

$$\log L = \sum_{i=1}^n \log P(y_i|x_i) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - f(x_i))^2$$

（$\log(2\pi\sigma^2)$ 为常数，忽略）

而 $MSE = \frac{1}{n}\sum_{i=1}^n (y_i - f(x_i))^2$，两者等价

• A2.4 (5 points) 尝试推导：在分类问题中，最小化交叉熵损失等价于极大化似然。



• A2.5 (5 points) 分析为什么L1正则化倾向于得到稀疏解、为什么L2正则化倾向于得到平滑的解。

在回归问题中，目标是最小化损失函数加个正则化项来控制模型复杂度。一般形式为目标函数=损失函数+λ·R(w),其中:

- w是模型的参数向量。
- R(w)是正则化项。
- λ是正则化强度的超参数。

对应的公式为

$$R_1(w) = \| w \|_1 = \sum | w_i |$$

$$R_2(w) = \| w \|_2 = \sum w_2{}^2$$

在参数优化过程中，L1和L2正则化会影响解的选择方式。在二维空间中，L1正则化的等高线是菱形。当最小化损失函数和正则化项的目标函数时，解位于损失函数的等高线与正则化项的等高线的切点处。由于L1正则化的等高线在坐标轴上有尖角，因此解更可能出现在坐标轴上，即某些参数为零，从而产生稀疏解。

正则化项会限制参数向量的范数。限制参数向量在一个L1球内。由于L1球在轴上有尖角，优化过程中，损失函数等高线更容易在这些尖角处与L1球相切。这些尖角对应于参数中的某些维度为零，从而导致参数向量的稀疏性。

与L1正则化不同，L2正则化的等高线是圆形或椭圆形，且在所有方向上都是平滑的，没有尖角。因此，优化过程中，损失函数等高线与L2球的切点更可能出现在轴外的地方，而不是坐标轴上。

我们还可以从梯度上解释。

L1 正则化的梯度:

$$\frac{\partial R_{\mathrm{L1}}}{\partial w_i} = sign(w_i)$$

当参数接近零时，L1正则化对参数的梯度是常数（+1或-1），这意味着即使参数值很小，仍然有力地将其推向零。

L2 正则化的梯度:

$$\frac{\partial R_{\mathrm{L2}}}{\partial w_i} = 2w_i$$

L2正则化的梯度与参数值成正比。当参数接近零时，梯度也变小，导致对参数的推力减弱。

也就是说，由于L1正则化在参数接近零时仍然有强大的推动力将参数变为零，因此更容易产生稀疏解。而L2正则化在参数接近零时推动力较小，使得参数更倾向于均匀地分布，而不是集中在少数参数上。

• A2.6 (5 points) 分析Batch normalization对参数优化起到什么作用、如何起到这种作用。

起到什么作用：

• 神经网络学习过程本质上就是为了学习数据分布，如果训练数据与测试数据的分布不同，网络的泛化能力就会严重降低。

• 输入层的数据，假设已经归一化，输入到后面，那么后面网络每一层的输入数据的分布一直在发生变化，前面层训练参数的更新将导致后面层输入数据分布的变化，必然会引起后面每一层输入数据分布的改变，网络前面几层微小的改变，后面几层就会逐步把这种改变累积放大。

BN的提出，就是要解决在训练过程中，中间层数据分布发生改变的情况。它可以在参数优化的时候提高模型的稳定性和泛化能力，防止梯度消失和爆炸的问题，也可以显著提高训练速度和收敛速度。

如何起到作用：

BN对每一层的输入进行标准化，使其具有零均值和单位方差。这一操作使得每一层的输入分布更加统一，减少了不同层之间输入分布差异带来的问题。还引入了可学习的缩放（γ）和偏移（β）参数，使得模型可以恢复甚至超过未标准化情况下的表示能力。

关于训练效率，通过对每一层的输入进行标准化，BN固定了输入的均值和方差，减少了网络中各层之间因输入分布变化而带来的干扰。这使得每一层可以更专注于学习任务，而不必应对输入分布的变化，从而提高了训练效率和稳定性。

关于梯度消失和爆炸，由于BN限制了每一层的输入范围，梯度在反向传播过程中也被规范化，使得梯度更加均衡地传播到各层，促进了有效的参数更新。

不过，BN目前仍然是一个黑盒，论文中有一定的理论推导，但是普遍还是通过对比提升来验证BN的效果。