梯度下降

梯度下降（Gradient Descent）是一种用于寻找函数局部最小值的一阶迭代优化算法。在机器学习和深度学习中，梯度下降被广泛用于优化损失函数，即调整模型参数以最小化损失函数的值。
梯度下降的基本思想是：**首先选择一个初始点作为起始点，然后在每一步迭代中，沿着函数梯度的反方向（即下降最快的方向）更新点的位置，直到达到一个局部最小值。**
梯度下降的更新公式为：

$\theta_{new} = \theta_{old} - \alpha \bigtriangledown_\theta J(\theta)$

其中：

- θ是模型参数。
- $J(\theta)$是损失函数。
- $\nabla_\theta J(\theta)$是损失函数相对于参数 θ的梯度。
- α 是学习率，控制了在梯度方向上的步长大小。

具体来说，损失函数的计算依赖于问题的类型（例如，回归或分类）、数据和模型的选择。例如，在线性回归问题中，常用的损失函数是均方误差（Mean Squared Error, MSE），它的形式如下：
$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
其中，m 是样本数量，$x^{(i)}$ 是第 i 个样本的特征向量，$y^{(i)}$ 是第 i 个样本的实际标签，$h_\theta(x)$ 是模型的预测输出。

为了计算梯度 $\nabla_\theta J(\theta)$，我们需要对损失函数 $J(\theta)$ 关于每个参数 θ_j 进行偏导：
$\frac{\partial}{\partial\theta_j}J(\theta)$
这个偏导数告诉我们当参数 θ_j 变化时，损失函数是如何变化的。

在梯度下降的每一步，我们计算当前参数 θ 下损失函数的梯度，然后更新 θ 来减少 $J(\theta)$。学习率 α 决定了更新的步长大小，如果太大可能会导致超过最小值，如果太小可能会使得学习过程非常慢。

损失函数就是相当于预测值与样本标签之间的差值的平均方差，根据上面的式子来得到新的θ，以此类推直至找到梯度为零的那个点，这个时候损失函数也最小。

主流方法比较

批量梯度下降（Batch Gradient Descent）：

- **优点：** 在处理整个训练集时计算效率高，通常能平滑地收敛到全局最小值。
- **缺点：** **如果数据集非常大，可能会变得缓慢且计算成本高**，因为需要将整个数据集加载到内存中。此外，由于缺乏噪声，它**可能会陷入局部最小值。**

随机梯度下降（Stochastic Gradient Descent, SGD）：

- **优点：** 更快且内存效率更高，因为它为每个训练样本更新参数。这对大型数据集非常有利，并且可以通过噪声帮助逃离局部最小值。
- **缺点：** 频繁更新可能导致训练过程中出现显著的波动和噪声，使收敛变得不稳定。

小批量梯度下降（Mini-Batch Gradient Descent）：

- **优点：** 通过为训练数据的子集更新参数，平衡了批量梯度下降和随机梯度下降的优点。这导致比批量梯度下降更频繁的更新，但比SGD具有**更少的噪声**。它在计算上更高效，更适合现代并行硬件。
- **缺点：** **引入了一个额外的超参数 - 小批量大小 - 需要进行优化**。也可能根据选择的批量大小产生复杂的函数。