优化算法介绍

前面的部分我们实际上已经介绍了梯度下降法来优化我们的模型，同时也已经开始用这个优化算法进行不断迭代来更新模型的参数，但是我们还不太明白优化算法的本质，在这一章，我们将具体介绍优化算法的原理。

优化算法对于深度学习非常重要，首先，实际训练一个模型所耗费的时间非常久，所以一旦选用了错误的优化算法，不仅影响训练的效率，同时也对模型的结果有着较大的影响；其次，如果我们深入理解了各种优化算法的本质，这也利于我们更有针对性地调参，从而使模型表现更好。

优化与机器学习

从前面讲的模型我们知道，一个机器学习的问题，我们会预先定义一个损失函数，然后通过最小化这个损失函数来优化我们的模型，而之所以只考虑最小化是因为任何一个最大化问题都能够转化成最小化问题，只需要在最大化问题前面增加一个负号，所以我们只需要关注如何最小化一个损失函数。

优化问题的挑战

对于一个简单函数的优化相对来说是很简单的，但是对于深度学习而言，优化往往是非常困难的，因为深度学习中的模型参数庞大，模型复杂，这意味着对于损失函数进行优化往往没有显示解（解析解），而需要使用基于数值方法的优化算法来找到近似解。这类优化问题基本上是目前深度学习的主流，都是基于不断地迭代来找到最优解，这就会导致两个问题，第一个就是求得一个较优的解所需要的时间相对较长，同时非常有可能找到局部最优解而非全局最优解。

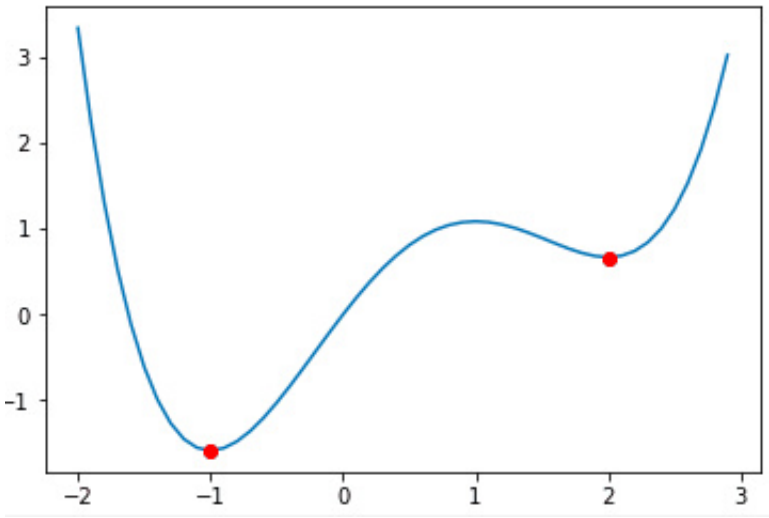
下面我们举两个局部最优解的例子，一个是局部最小点，一个是鞍点。

局部最小点

对于一个函数来讲，全局最小点就是这个点在整个函数的定义域中都是最小的，而局部最小点就是这个点在某个局部是最小点，比如

$$f(x) = 2x - 0.5x^2 - \frac{2}{3}x^3 + \frac{1}{4}x^4 \tag{1}$$

其 (-2, 3) 之间的图像如下，可以看到 -1 是他的最小值点，而 2 是他的局部最小值点



因为梯度下降法是基于损失函数的梯度进行参数更新，如果更新到局部最小点，梯度就为 0，就存在陷入局部极小点的情况

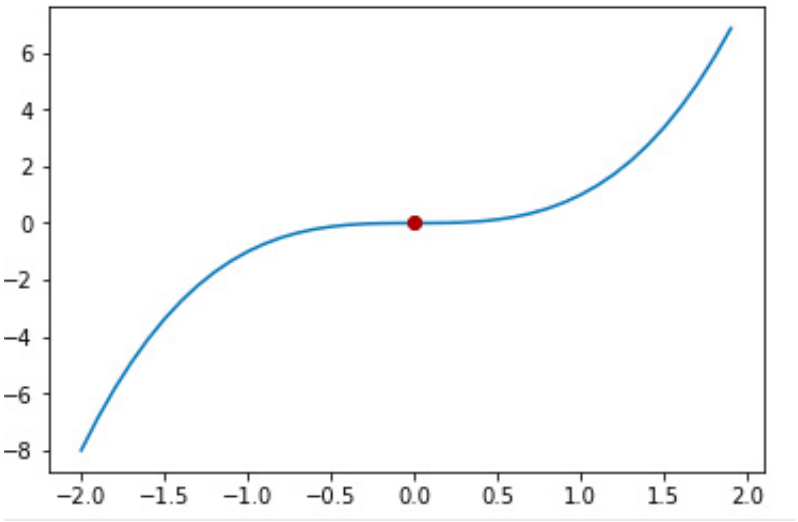
鞍点

除了局部极小点的梯度为 0 之外，还有另外一类点的梯度也为 0，就是鞍点，但是鞍点并不是周围所有点中最小的点，只是梯度为 0 使得模型难以更新所导致的情况。在深度学习中，因为损失函数非常复杂，所以基本上碰到的都是鞍点。

比如

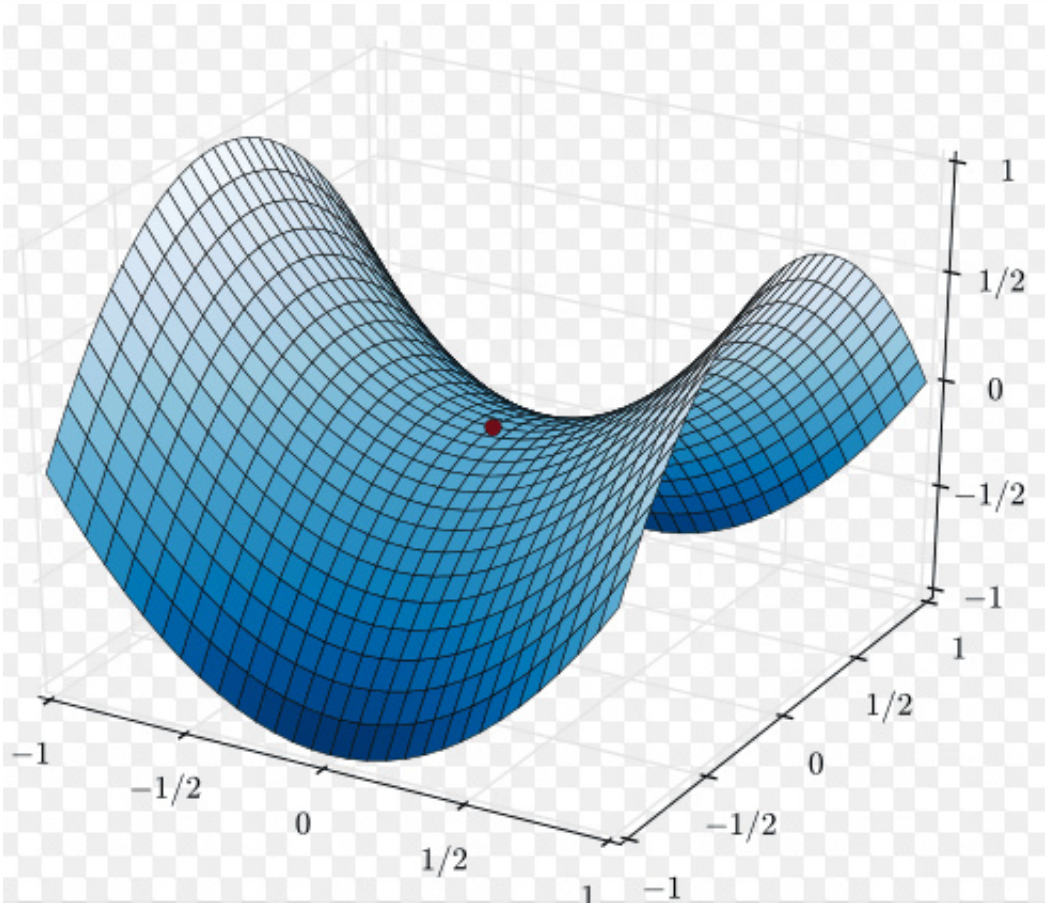
$$y = x^3 \tag{2}$$

图像如下



其在 x = 0 的地方梯度为 0，是一个鞍点，但是其比它左边的值都大

另外一个二维平面鞍点的例子如下



红色的点所在的位置就是鞍点，这个地方的梯度为 0，如果模型陷入鞍点，往往无法得到较好的结果

在深度学习中要找到全局最优解是非常非常困难的，但是这并不是必要的，因为实际问题中一个较优的解已经能够比较好的解决问题，下面我们具体讲一讲几种不同的优化算法。

随机梯度下降

随机梯度下降法是我们前面一直在使用的优化算法，但是我们使用下山的例子来说明梯度下降法的原理，就是每次都选取下降最快的方向，下面我们说明一下梯度下降法的数学原理。

对于损失函数 $L(\theta)$，我们使用其梯度 $\nabla L(\theta)$ 来更新参数 θ，学习率为 η，那么更新公式为

$$\theta^i = \theta^{i-1} - \eta \nabla L(\theta^{i-1}) \tag{3}$$

我们重新表述一下我们的问题，我们希望求解下面这个方程

$$\theta^* = \arg\min_{\theta} L(\theta) \tag{4}$$

也就是说我们希望更新参数之后有下面的结果

$$L(\theta_0) > L(\theta_1) > L(\theta_2) > \cdots \tag{5}$$

在解决这个问题之前，我们需要先回顾一下泰勒级数。

泰勒级数

对于任何一个无限可微函数 $h(x)$，在一个点 $x = x_0$ 附近，我们有以下的泰勒级数

$$h(x) = \sum_{k=0}^{\infty} \frac{h^{(k)}(x_0)}{k!}(x - x_0)^k \tag{6}$$

$$= h(x_0) + h^{'}(x_0)(x - x_0) + \frac{h^{''}(x_0)}{2!}(x - x_0)^2 + \cdots \tag{7}$$

当 x 足够接近 x_0，我们有下面的近似

$$h(x) \approx h(x_0) + h^{'}(x_0)(x - x_0) \tag{8}$$

对于多元泰勒级数，我们有以下的公式

$$h(x, y) = h(x_0, y_0) + \frac{\partial h(x_0, y_0)}{\partial x}(x - x_0) + \frac{\partial h(x_0, y_0)}{\partial y}(y - y_0) + \cdots \tag{9}$$

同样当x和y足够接近 x_0 和 y_0 的时候，我们有以下的近似

$$h(x, y) \approx h(x_0, y_0) + \frac{\partial h(x_0, y_0)}{\partial x}(x - x_0) + \frac{\partial h(x_0, y_0)}{\partial y}(y - y_0) \tag{10}$$

一维情况的简单说明

对于损失函数 L，目前的参数是 θ_i，我们对其在 θ_i 处进行 taylor 展开可以得到下面的结果

$$L(\theta_i + \epsilon) \approx L(\theta_i) + L'(\theta_i)\epsilon \tag{11}$$

如果学习率 η 非常小，那么每次的参数更新量 $-\eta L'(\theta_i)$ 就非常小，我们可以将其带入上面的式子，得到

$$L(\theta_i - \eta L'(\theta_i)) \approx L(\theta_i) - \eta L'(\theta_i)^2 \tag{12}$$

因为 η 是一个很小的正数，而 $L'(\theta_i)^2$ 也是一个正数，所以就有

$$L(\theta_i - \eta L'(\theta_i) \leq L(\theta) \tag{13}$$

通过上面的例子说明，我们确实论证了每次参数更新，损失函数都在不断减少。

虽然都有了数学的证明，但是为什么我们训练的时候损失函数并不是一直减少，而是一个不断上下跳动，慢慢减小的过程呢？一是因为 $\eta L'(\theta_i)$ 并不一定是一个特别小的量，所以不能完美的适用于 taylor 公式，第二点就是我们一般使用的叫做随机梯度下降，也就是每次取一定量的数据集来计算梯度，而不是全部的数据集，因为全部的数据集太过于庞大，而且计算非常慢，随机取数据点不但计算快，还有利于我们跳出局部极小点和鞍点。

二维情况的证明

现在假设我们的参数 θ 有两个变量 θ_1, θ_2，那么由上面的二元泰勒级数我们知道，对于一个点(a, b)，对于一个足够小的范围，我们有以下近似

$$L(\theta) = L(a,b) + \frac{\partial L(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial L(a,b)}{\partial \theta_2}(\theta_2 - b) \tag{14}$$

我们令 $s = L(a,b), u = \frac{\partial L(a,b)}{\partial \theta_1}, v = \frac{\partial L(a,b)}{\partial \theta_2}$，那么$L(\theta)$ 有以下的简单表达

$$L(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b) \tag{15}$$

我们知道s，u和v都是常数，所以我们希望找到 θ_1, θ_2 使得 $L(\theta)$ 最小，同时 θ_1, θ_2 都是在(a, b)的一个小范围之内，所以要满足以下条件

$$(\theta_1 - a)^2 + (\theta_2 - b)^2 \leq \epsilon^2 \tag{16}$$

我们再换元，将 $(\theta_1 - a) = \Delta\theta_1, (\theta_2 - b) = \Delta\theta_2$，同时由于s是一个与 θ_1, θ_2 没有关系的常量，所以我们可以去掉它，这样我们就能够得到下面的式子

$$L(\theta) \Leftrightarrow u\Delta\theta_1 + v\Delta\theta_2 \tag{17}$$

$$\Delta\theta^2 + \Delta\theta^2 \leq \epsilon^2 \tag{18}$$

对于上面这个式子，我们可以将(u, v)看作一个向量，同时 $(\Delta\theta_1, \Delta\theta_2)$ 也是一个向量，所以 $(u, v) \cdot (\Delta\theta_1, \Delta\theta_2) = u\Delta\theta_1 + v\Delta\theta_2$。那么怎么能够求到这个内积的最小值呢，很简单，我们只需要 $(\Delta\theta_1, \Delta\theta_2)$ 是(u, v)的反方向，也就是 $(\Delta\theta_1, \Delta\theta_2) = -(u, v)$，但是由于我们需要限制 $\Delta\theta_1, \Delta\theta_2$ 都在一个小的范围内，所以要对他们的范数做一个限制，也就是

$$(\Delta\theta_1, \Delta\theta_2) = -\eta(u, v) \tag{19}$$

$$\Longleftrightarrow (\theta_1, \theta_2) = (a, b) - \eta(u, v) \tag{20}$$

所以只要在一个足够小的范围内，我们需要参数更新之后取得一个更小的值，那么我们有以下的更新公式

$$(\theta_1, \theta_2) = (a, b) - \eta(\frac{\partial L(a,b)}{\partial \theta_1}, \frac{\partial L(a,b)}{\partial \theta_2}) \tag{21}$$

所以只要我们取一个特别小的学习率 η，我们就能够保证 θ_1, θ_2 在一个足够靠近(a, b)的范围内，这就实现了我们之前讲的梯度下降法。

我们再回顾一下随机梯度下降法的公式

$$\theta_{i+1} = \theta_i - \eta\nabla L(\theta) \tag{22}$$