批量梯度

1. 什么是梯度下降（Gradient Descent）？

梯度下降是一种**优化算法**，用于最小化目标函数（如机器学习中的损失函数 $J(\theta)$）。其核心思想是：

> **沿函数梯度（一阶导数）的反方向迭代调整参数，逐步逼近函数最小值。**

- **类比**：想象你在山顶蒙眼下山。每步沿最陡峭的下坡方向走，最终会到达山谷（最小值点）。
- **关键概念**：
  - **梯度**：函数在各参数方向上的偏导数向量，指向函数增长最快的方向。
  - **学习率（Learning Rate）** α：控制每次参数更新的步长。

2. 梯度下降的数学推导

问题设定：

- 损失函数： $J(\theta)$ （ θ 为模型参数向量，例如线性回归中的权重）
- 目标：找到 θ^* 使得 $J(\theta)$ 最小化。

推导步骤：

1. 泰勒展开近似

在 θ 的邻域内， $J(\theta)$ 可近似为：

$$J(\theta + \Delta\theta) \approx J(\theta) + \nabla J(\theta)^T \Delta\theta$$

其中 $\nabla J(\theta)$ 是梯度（偏导数向量）。

2. 参数更新方向

为使 $J(\theta + \Delta\theta) < J(\theta)$，需满足：

$$\nabla J(\theta)^T \Delta\theta < 0$$

当 $\Delta\theta = -\alpha \nabla J(\theta) (\alpha > 0)$ 时：

$$\nabla J(\theta)^T (-\alpha \nabla J(\theta)) = -\alpha \|\nabla J(\theta)\|^2 < 0$$

看到其 沿负梯度方向更新可保证函数值下降。

3. 参数更新规则

最终我们得到：

$$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \cdot \nabla J(\theta_{\text{old}})$$

3. 梯度计算推导

线性回归损失函数对参数 θ_0 和 θ_1 偏导数的计算过程。从损失函数的定义出发，逐步推导出最终公式。

1. 损失函数定义

假设函数： $h_\theta(x) = \theta_0 + \theta_1 x$

损失函数（MSE加个 $\frac{1}{2}$ ，别问为什么加，问就是好导）：

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

2. 展开假设函数

将 $h_\theta(x^{(i)})$ 代入：

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

3. 求对 θ_0 的偏导数（ $\frac{\partial J}{\partial \theta_0}$ ）

使用链式法则和求和符号的导数性质：

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_0} \left[(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2 \right]$$

对内部函数求导（设 $u_i = \theta_0 + \theta_1 x^{(i)} - y^{(i)}$，则 $\frac{\partial}{\partial \theta_0}(u_i^2) = 2u_i \cdot \frac{\partial u_i}{\partial \theta_0}$，且 $\frac{\partial u_i}{\partial \theta_0} = 1$ ）：

$$\frac{\partial}{\partial \theta_0} \left[(\cdots)^2 \right] = 2(\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot 1$$

代入原式：

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{2m} \sum_{i=1}^{m} 2(\theta_0 + \theta_1 x^{(i)} - y^{(i)}) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

由于 $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$ ，最终简化为：

$$\boxed{\frac{\partial J}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}$$

4. 求对 θ_1 的偏导数（ $\frac{\partial J}{\partial \theta_1}$ ）

类似地：

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial \theta_1}\left[(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2\right]$$

对内部函数求导（ $\frac{\partial u_i}{\partial \theta_1} = x^{(i)}$ ）：

$$\frac{\partial}{\partial \theta_1}\left[(\cdots)^2\right] = 2(\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

代入原式：

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{2m}\sum_{i=1}^{m}2(\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot x^{(i)} = \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

最终简化为：

$$\boxed{\frac{\partial J}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}}$$

4. 批量梯度下降（Batch Gradient Descent）

定义：

> 每次迭代使用全部训练样本计算损失函数的梯度。

算法步骤：

1. 初始化参数 θ （如随机向量）
2. 重复直至收敛：
   - 计算梯度： $\nabla J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$
   - 更新参数： $\theta := \theta - \alpha \cdot \nabla J(\theta)$ （其中 m 为样本总数）

5. 数学示例（线性回归，最终梯度公式版本）

1. 上述公式总结

模型：

$$h_\theta(x) = \theta_0 + \theta_1 x$$

损失函数（MSE）：

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

梯度计算：

$$\begin{cases} \frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\ \frac{\partial J}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \end{cases}$$

参数更新：

$$\begin{cases} \theta_0 := \theta_0 - \alpha \cdot \frac{\partial J}{\partial \theta_0} \\ \theta_1 := \theta_1 - \alpha \cdot \frac{\partial J}{\partial \theta_1} \end{cases}$$

2. 问题设置

假设我们有一个线性回归模型，其假设函数为：

$$h_\theta(x) = \theta_0 + \theta_1 x$$

其中 θ_0 和 θ_1 是待学习的参数。

损失函数采用均方误差（MSE），但为了简化梯度计算，我们使用以下形式（引入常数 $\frac{1}{2}$ 以方便求导）：

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

其中：

- m 是训练样本的数量。
- $(x^{(i)}, y^{(i)})$ 是第 i 个训练样本。

3. 示例数据集

为了简单起见，我使用一个小型数据集进行手动计算：

- 样本数量 $m = 3$。
- 训练样本：
  - 样本 1: $(x^{(1)}, y^{(1)}) = (1, 2)$
  - 样本 2: $(x^{(2)}, y^{(2)}) = (2, 3)$
  - 样本 3: $(x^{(3)}, y^{(3)}) = (3, 4)$

4. 初始参数和学习率

- 初始参数： $\theta_0 = 0, \theta_1 = 0$ （故意设置为零以简化计算）。
- 学习率 或者说 步长： $\alpha = 0.1$ （一个常见的选择）。

5. 迭代过程（第一次迭代）

我们将执行一次完整的批量梯度下降迭代。关键在于：梯度计算使用整个数据集。

步骤 1: 计算当前假设和误差

使用当前参数 ($\theta_0 = 0, \theta_1 = 0$) 计算每个样本的预测值和误差：

- 对于样本 1: $h_\theta(x^{(1)}) = \theta_0 + \theta_1 \cdot x^{(1)} = 0 + 0 \cdot 1 = 0$，误差 $= h_\theta(x^{(1)}) - y^{(1)} = 0 - 2 = -2$
- 对于样本 2: $h_\theta(x^{(2)}) = 0 + 0 \cdot 2 = 0$ ，误差 = $0 - 3 = -3$
- 对于样本 3: $h_\theta(x^{(3)}) = 0 + 0 \cdot 3 = 0$ ，误差 = $0 - 4 = -4$

步骤 2: 计算梯度（使用整个数据集）

计算损失函数对每个参数的偏导数（梯度）：

- 对于 θ_0 ：

$$\begin{aligned} \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) && (1) \\ &= \frac{1}{3} \left[(-2) + (-3) + (-4) \right] && (2) \\ &= \frac{1}{3} \times (-9) = -3 && (3) \end{aligned}$$

- 对于 θ_1 ：

$$\begin{aligned} \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} && (4) \\ &= \frac{1}{3} \left[(-2) \cdot 1 + (-3) \cdot 2 + (-4) \cdot 3 \right] && (5) \\ &= \frac{1}{3} \left[-2 - 6 - 12 \right] && (6) \\ &= \frac{1}{3} \times (-20) && (7) \\ &= -\frac{20}{3} \approx -6.6667 && (8) \end{aligned}$$

步骤 3: 更新参数

使用梯度下降更新规则：

- 更新 θ_0 ：

$$\theta_0^{\text{new}} = \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J = 0 - 0.1 \times (-3) = 0 + 0.3 = 0.3$$

- 更新 θ_1：

$$\theta_1^{\text{new}} = \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J = 0 - 0.1 \times \left(-\frac{20}{3}\right) = 0 + 0.1 \times \frac{20}{3} = \frac{2}{3} \approx 0.6667$$

第一次迭代后的结果

- 新参数： $\theta_0 \approx 0.3$, $\theta_1 \approx 0.6667$
- 损失函数值的变化（可选计算）：初始损失 $J(0,0) = \frac{1}{2 \times 3} \sum [(-2)^2 + (-3)^2 + (-4)^2] = \frac{1}{6} \times 29 \approx 4.8333$ ，新损失 $J(0.3, 0.6667) \approx 1.7556$ （计算略），损失减小，表明优化有效。

6. 关键问题与注意事项

1. **学习率选择**：
   - α 过小 → 收敛慢； α 过大 → 震荡甚至发散。
   - **建议**：尝试 $\alpha = 0.001, 0.01, 0.1$ ，或用学习率衰减策略。
2. **收敛判断**：
   - 停止条件： $\|\theta_{\text{new}} - \theta_{\text{old}}\| < \varepsilon$ 或 $\|J(\theta_{\text{new}}) - J(\theta_{\text{old}})\| < \varepsilon$ 。
3. **特征缩放**：
   - 若特征量纲差异大（如 $x_1 \in [0,1], x_2 \in [0,1000]$ ），需归一化以加速收敛。
4. **批量 vs 随机 vs 小批量**：

方法	计算梯度所用数据	速度	稳定性
批量梯度下降	全数据集	慢	高
随机梯度下降（SGD）	单样本	快	低（震荡）
小批量梯度下降	小批量（如32）	中等	中等

5. **梯度方向稳定，收敛到全局最优（凸函数）**
6. **理论保证收敛**
7. **每次计算需遍历全数据集，速度慢（随机和少量不会）**

7. 精华总结

- **梯度下降**：通过负梯度方向迭代优化参数的通用框架。
- **批量梯度下降**：使用全数据计算真梯度，收敛稳但计算成本高。
- **核心公式**： $\theta := \theta - \alpha \cdot \nabla_\theta J(\theta)$
- **为什么标题是批量梯度，但推导是批量梯度下降**：只需要变学习率前的符号即可，直接变上升
- **实际上还有两个**：随机梯度下降和少量批量梯度下降，但是这里不讨论，未来再说

> **附：Python函数伪代码（记得导包）**

代码块

```
def batch_gradient_descent(X, y, alpha, epochs):
    m, n = X.shape
    theta = np.zeros(n)  # 初始化参数
    for _ in range(epochs):
        grad = (1/m) * X.T.dot(X.dot(theta) - y)  # 计算梯度（线性回归）
        theta -= alpha * grad  # 更新参数
    return theta
```