调研不同类型的梯度下降法

你的名字和学号

2025 年 6 月 23 日

一、牛顿法：

牛顿法，是一种用于求解方程根或函数的最小值的迭代优化算法。它利用函数的二阶导数信息（Hessian 矩阵）来逼近函数的局部性质，能够更快地收敛到最优解。

牛顿法的基本思想是通过构造函数的泰勒级数展开来近似原函数，并使用近似函数的根或最小值来逐步逼近原函数的根或最小值。在每一次迭代中，牛顿法使用当前点的切线来估计函数的根或最小值，并将切线与 x 轴的交点作为下一次迭代的点。这样，通过不断迭代，可以逐渐逼近函数的根或最小值。

在现有的极小值估计值的附近对 f(x) 做二阶泰勒展开，进而找到极小点的下一个估计值，反复迭代直到函数的一阶导数小于某个接近 0 的阀值。最终求出极小点的估计值。

牛顿法的基本更新公式

牛顿法的参数更新公式为：

$$\theta_{t+1} = \theta_t - H^{-1}(\theta_t)\nabla f(\theta_t) \tag{1}$$

其中:

1. θ_t：第 t 次迭代的参数值
2. $\nabla f(\theta_t)$：目标函数在 θ_t 处的梯度（一阶导数向量）
3. $H(\theta_t)$：Hessian 矩阵（二阶导数矩阵），定义为：

$$H_{ij}(\theta) = \frac{\partial^2 f}{\partial\theta_i \partial\theta_j} \tag{2}$$

4. $H^{-1}(\theta_t)$：Hessian 矩阵的逆矩阵

扩展说明

当 Hessian 非正定时，常使用修正形式:

$$\theta_{t+1} = \theta_t - (H(\theta_t) + \lambda I)^{-1}\nabla f(\theta_t) \tag{3}$$

牛顿法的作用：

通过利用函数的二阶导数信息，牛顿法能够更快地收敛到函数的最优解。在优化问题中，牛顿法常用于求解无约束优化问题或约束优化问题的局部最优解。它在数学建模、机器学习、数据分析等领域中具有重要应用，例如最小二乘法、逻辑回归、神经网络等模型的参数优化。

二、共轭梯度法：

核心原理：

共轭梯度法是一种迭代算法，用于求解对称正定矩阵的线性方程组 $A\mathbf{x} = \mathbf{b}$。通过构造一组*共轭方向*（满足 $\mathbf{d}_i^\top A\mathbf{d}_j = 0\ (i \neq j)$），确保每次迭代沿新方向更新解时不会破坏之前的优化结果。

算法公式：

1. **初始化：**

$$\mathbf{x}_0 = \text{初始解估计} \tag{4}$$

$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0 \quad \text{(初始残差)} \tag{5}$$

$$\mathbf{d}_0 = \mathbf{r}_0 \quad \text{(初始搜索方向)} \tag{6}$$

2. **迭代步骤**（第 k 步）：

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{d}_k^\top A\mathbf{d}_k} \quad \text{(步长计算)} \tag{7}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \quad \text{(解更新)} \tag{8}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{d}_k \quad \text{(残差更新)} \tag{9}$$

$$\beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k} \quad \text{(Fletcher-Reeves 系数)} \tag{10}$$

$$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k \quad \text{(方向更新)} \tag{11}$$

3. **终止条件**：当 $\|\mathbf{r}_k\| < \epsilon$ 时停止。

特性：

1. **有限步收敛**: 对 n 维问题，理论最多需 n 步（实际受数值误差影响）
2. **内存高效**: 仅需存储 $\mathbf{x}, \mathbf{r}, \mathbf{d}, A\mathbf{d}$ 四个向量
3. **适用场景**: 大规模稀疏矩阵、二次型优化问题

扩展形式：对非线性目标函数 $f(\mathbf{x})$：

- 残差定义为负梯度：$\mathbf{r}_k = -\nabla f(\mathbf{x}_k)$
- 使用修正系数（如 Polak-Ribiére 公式）：

$$\beta_k^{\text{PR}} = \frac{\mathbf{r}_{k+1}^\top (\mathbf{r}_{k+1} - \mathbf{r}_k)}{\mathbf{r}_k^\top \mathbf{r}_k}$$

- 定期重启（重置 $\mathbf{d}_k = \mathbf{r}_k$）

三、SGD：

SGD 的核心思想是在每次迭代中随机选择一个样本（或一小批样本）来估计梯度，而不是使用整个数据集。这样做的优点是计算效率高，尤其是当数据集很大时。SGD 也能够逃离局部最小值，因为随机性引入了一定的噪声，有助于模型探索更多的参数空间。

SGD 更新公式如下

$$\theta_{t+1} = \theta_t - \eta \nabla f_i(\theta_t)$$

其中：

θ 表示模型参数

t 表示当前迭代次数

η 表示学习率，控制步长大小

fi 是损失函数，在每次迭代中，我们计算当前 θ_t 的梯度，然后按照梯度反方向更新参数，以减小损失函数的值

四、动量法：

动量法（Momentum）是一种优化算法，旨在加速梯度下降法的收敛，尤其是在存在高曲率、嘈杂梯度或小但一致梯度的情况下。动量法通过引入动量概念，使得参数更新不仅依赖于当前的梯度，还考虑了之前梯度的累计效果，从而加速收敛并减少参数更新时的震荡。

动量法的原理：

动量法的核心思想是引入一个动量项（velocity），记录之前的梯度信息，并在每次参数更新时加上这个动量项。具体来说，动量法的更新公式如下：

$$v_t = \beta v_{t-1} + (1 - \beta) \nabla_\theta J(\theta)$$

其中：

1. v_t 是第 t 次迭代的动量项；

2. β 是动量超参数，通常取值在 0 到 1 之间，控制之前梯度的影响程度；

3. $\nabla_\theta J(\theta)$ 是损失函数 $J(\theta)$ 对参数 θ 的梯度。

参数更新：

$$\theta = \theta - \alpha v_t$$

其中：

1. θ 是模型参数；

2. α 是学习率，控制每次更新的步长。

动量法通过引入动量项，使得参数更新不仅依赖于当前的梯度，还考虑了之前梯度的累积效果，从而加速收敛并减少参数更新时的震荡。

五、ADAM：

ADAM 结合了动量法（Momentum）和 RMSProp 的思想，旨在通过计算梯度的一阶矩估计和二阶矩估计来调整每个参数的学习率，从而实现更高效的网络训练。

Adam 算法的关键在于同时计算梯度的一阶矩（均值）和二阶矩（未中心的方差）的指数移动平均，并对它们进行偏差校正，以确保在训练初期时梯度估计不会偏向于 0。

Algorithm 1 Adam 优化算法

初始化参数 θ_0，一阶矩 $m_0 = 0$，二阶矩 $v_0 = 0$，时间步 $k = 1$
while 未收敛 **do**
  计算梯度：$g = \nabla_{\theta_{k-1}} L(\theta)$
  更新一阶矩：$m_k = \beta_1 m_{k-1} + (1 - \beta_1) g$
  更新二阶矩：$v_k = \beta_2 v_{k-1} + (1 - \beta_2) g \odot g$
  偏差校正一阶矩：$\hat{m}_k = \frac{m_k}{1 - \beta_1^k}$
  偏差校正二阶矩：$\hat{v}_k = \frac{v_k}{1 - \beta_2^k}$
  更新参数：$\theta_k = \theta_{k-1} - \frac{\eta}{\sqrt{\hat{v}_k} + \epsilon} \hat{m}_k$
  $k \leftarrow k + 1$
end while

参数说明：

1. m_k 和 v_k：梯度的一阶矩和二阶矩估计；
2. β_1, β_2：指数衰减率，通常设为 0.9 和 0.999；
3. ϵ：极小值（如 10^{-8}），防止除以零；
4. k：当前迭代次数；
5. β_1^k 和 β_2^k：定义为 β_1 和 β_2 的 k 次连乘，用于偏差校正。

偏差校正的作用：

1. **对 m_k 的偏差校正**：初始时刻，一阶矩 m_k 的值偏小，因为它是梯度值的加权平均，起始所有梯度都被初始化为 0。通过除以 $1 - \beta_1^k$，可以将 m_k 的值放大，使其更快地接近实际的梯度均值。随着迭代次数 k 的增加，β_1^k 会趋向于 0，偏差校正因子 $1 - \beta_1^k$ 趋向于 1，校正影响逐渐减小。

数学表达：

$$\hat{m}_k = \frac{m_k}{1 - \beta_1^k} \quad \text{其中} \quad \beta_1^k = \prod_{i=1}^{k} \beta_1$$

2. **对 v_k 的偏差校正**：类似地，二阶矩 v_k（梯度平方的加权平均）在初始阶段被低估。通过除以 $1 - \beta_2^k$，可以增加 v_k 的值，使其更接近实际的梯度平方均值。随着 k 增加，偏差校正因子 $1 - \beta_2^k$ 也趋向于 1。

数学表达：

$$\hat{v}_k = \frac{v_k}{1 - \beta_2^k} \quad \text{其中} \quad \beta_2^k = \prod_{i=1}^{k} \beta_2$$

对比总结与选型建议

算法	核心优势	局限性	典型应用
共轭梯度	内存高效，二次问题收敛快	仅适合中小规模问题	物理模拟、线性方程组求解
牛顿法	二阶收敛速度	计算成本高，易受鞍点影响	小规模凸优化（如金融模型）
SGD	灵活，适合超大数据	需调参，收敛慢	大规模深度学习初步训练
Momentum	减少震荡，加速收敛	可能越过最优解	需要稳定训练的视觉任务
Adam	自适应学习率，收敛快且稳定	可能泛化略逊于调优的 SGD	深度学习默认选择（如 Transf