

Deep Learning Technology and Application

Ge Li

Peking University

训练优化方法

牛顿法

设待最小的函数为： $f(x)$ ，即：求取 x 使 $f(x)$ 最小：

$$\min_x f(x).$$

【带 Peano 余项的 Taylor 公式】 设 $f(x)$ 在 x_k 处有 n 阶导数，则存在 x_k 的一个邻域，对于该邻域中的任一点 x ，成立：

$$\begin{aligned} f(x) = & f(x_k) + f'(x_k)(x - x_k) + \frac{f''(x_k)}{2!}(x - x_k)^2 + \dots \\ & + \frac{f^{(n)}(x_k)}{n!}(x - x_k)^n + o((x - x_k)^n) \end{aligned}$$

余项 $o((x - x_n)^n)$ 为高阶无穷小。

牛顿法

设 x_k 为当前的极小值估计值, 则在 x_k 邻域中的任一点 x 点做二阶泰勒展开, 得到:

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(x_k)}{2!}(x - x_k)^2$$

若在 x_k 邻域中的任一点 x 处能够取得最小值, 即: $f'(x) = 0$ 于是得:

$$f'(x_k) + f''(x_k)(x - x_k) = 0$$

于是得:

$$x = x_k - \frac{f'(x_k)}{f''(x_k)}$$

得到 x 的迭代更新规律为:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

牛顿法

将上式中的 x 推广至 N 维的情形 (下文中 x 表示 N 向量), 设 x_k 为第 k 步时, 与目标函数极小值的估计值所对应的 x , 则在 x_k 处目标函数 $f(x)$ 的二阶泰勒展开式为 :

$$f(x) = f(x_k) + \nabla f(x_k) \cdot (x - x_k) + \frac{1}{2} \cdot (x - x_k)^T \cdot \nabla^2 f(x_k) \cdot (x - x_k)$$

其中, ∇f 为 f 的梯度向量, $\nabla^2 f$ 为 Hessian 矩阵 :

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix} \text{ 记为: } g, \quad \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix} \text{ 记为: } H$$

注 : 仅在上式中, x_i 表示 N 维向量 x 的第 i 维 ;

牛顿法

所以上式变为：

$$f(x) = f(x_k) + g_k^T (x - x_k) + \frac{1}{2} \cdot (x - x_k)^T \cdot H_k \cdot (x - x_k)$$

其中, g_k 表示 $\nabla f(x_k)$, H_k 表示 $\nabla^2 f(x_k)$;

若在 x_k 邻域中的任一点 x 处能够取得最小值, 即: $f'(x) = 0$ 于是, 对上式对 x 求导数, 得:

$$g_k + H_k \cdot (x - x_k) = 0$$

若 H_k 非奇异, 则得到:

$$x = x_k - H_k^{-1} \cdot g_k$$

得到 x 的迭代更新规律为:

$$x_{k+1} = x_k - H_k^{-1} \cdot g_k$$

$d_k = -H_k^{-1} \cdot g_k$ 被称为“牛顿方向”.

阻尼牛顿法

- 当目标函数为二次函数时，Hessian 矩阵退化成一个常数矩阵，从任一初始点出发只需要一步迭代即可达到 $f(x)$ 的极小点 x^* ，这就是牛顿法的“二次收敛性”。
- 然而，牛顿法的迭代公式中，每步迭代都是固定长度，因此，并不一定能够收敛。对于非二次型目标函数，甚至可能会使函数值上升，导致计算失败。
- 为了克服这个问题，人们给牛顿法增加一个“步长因子” λ_k ，且该步长因子满足：

$$\lambda_k = \operatorname{argmin}_{\lambda \in \mathcal{R}} f(x_k + \lambda d_k)$$

- 其中，难点在于计算 $d_k = -H_k^{-1} \cdot g_k$ 。
- 迭代终止条件可以设定为： $\|g_k\| < \epsilon$

阻尼牛顿法

- ① Initialize: $k = 0$; $x_k = \text{random number}$; $\epsilon > 0$;
 - ② Calculate: g_k and H_k ;
 - ③ If $\|g_k\| < \epsilon$, then return x_k ;
 else calculate $d_k = -H_k^{-1} \cdot g_k$;
 - ④ Calculate: $\lambda_k = \operatorname{argmin}_{\lambda \in \mathcal{R}} f(x_k + \lambda d_k)$
 - ⑤ Calculate: $x_{k+1} = x_k + \lambda_k d_k$;
 - ⑥ Iterate: $k+ = 1$; goto step 2;
- 优点：利用目标函数的二阶导数，不但考虑了一阶方向性，且利用了二阶导数对变化趋势的预测能力；
 - 缺点：
 - 对目标函数的要求很高——要求二阶可导，且 Hessian 矩阵必须为正定矩阵；
 - 计算量大，需要求解 Hessian 矩阵的逆，计算复杂度为 $O(n^3)$ 级；

拟牛顿法的基本思想

基本思想：针对牛顿法存在的上述缺点，通过构造 Hessian 矩阵的近似矩阵的方法，对目标函数进行优化。希望该矩阵具有以下性质：

- 该矩阵应该能够具有逼近（模拟）目标函数二阶导数的特性；
- 该矩阵不必求取 Hessian 矩阵；
- 更不必求取 Hessian 矩阵的逆矩阵；
- **也许**，该矩阵应该具备递推性质，即由 k 情况下的矩阵值，可以递推出 $k + 1$ 情况下的矩阵值。即：

$$\hat{H}_{k+1} = \hat{H}_k + \Delta \hat{H}_k$$

如果存在一个这样的矩阵 \hat{H}_k ，那么原始牛顿法的计算过程，将改为：

拟牛顿法的计算过程

如上所述, 设 \hat{H} 为 Hessian 矩阵 H^{-1} 的同阶近似矩阵;

- ① Initialize: $k = 0$; x_k as random number; $\epsilon > 0$;
 \hat{H}_0 as positive definite and symmetric matrix;
- ② Calculate: g_k ;
- ③ If $\|g_k\| < \epsilon$, then return x_k ;
- ④ Calculate $g_{k+1} = \nabla f(x_k)$, $\hat{H}_{k+1} = \hat{H}_k + \Delta \hat{H}_k$;
- ⑤ Calculate $d_{k+1} = -\hat{H}_{k+1} \cdot g_{k+1}$;
- ⑥ Calculate: $\lambda_{k+1} = \operatorname{argmin}_{\lambda \in \mathcal{R}} f(x_k + \lambda d_{k+1})$;
- ⑦ Calculate: $x_{k+1} = x_k + \lambda_k d_{k+1}$;
- ⑧ Iterate: $k+ = 1$; goto step 2;

拟牛顿法的构造条件

在上述方法中，只有 \hat{H} 未知，因此，**现在的关键是：如何构造 \hat{H} 。**
 先来分析一下 \hat{H}_{k+1} 应该满足的条件：

- ① 条件 1：由上文可知，该矩阵**应该具备递推性质**，即由 k 情况下的矩阵值，可以递推出 $k+1$ 情况下的矩阵值。即：

$$\hat{H}_{k+1} = \hat{H}_k + \Delta \hat{H}_k$$

- ② 条件 2：由 $x_{k+1} = x_k + \lambda_k d_{k+1}$ 且 $d_{k+1} = -\hat{H}_{k+1} \cdot g_{k+1}$ 可知，为了确保 x 的搜索方向稳定，**最好保持 \hat{H}_{k+1} 为正定矩阵；**
- ③ **条件 3 在数学上，必须满足“拟牛顿条件”：**

拟牛顿条件

设当前要优化的目标函数为 $f(x)$ ，设已完成的迭代步骤为 k ；需要推知 $k+1$ 情况下的计算方法；

下面开始推导：

- ① 首先将 $f(x)$ 在 x_{k+1} 点展开：

$$\begin{aligned} f(x) &= f(x_{k+1}) + \nabla f(x_{k+1}) \cdot (x - x_{k+1}) \\ &\quad + \frac{1}{2} \cdot (x - x_{k+1})^T \cdot \nabla^2 f(x_{k+1}) \cdot (x - x_{k+1}) \end{aligned}$$

- ② 对两边关于 x 求导：

$$\nabla f(x) = \nabla f(x_{k+1}) + H_{k+1} \cdot (x - x_{k+1})$$

- ③ 在上式中，取 x 为当前已经迭代到的值 x_k ，得到：

$$\nabla f(x_k) - \nabla f(x_{k+1}) = H_{k+1} \cdot (x_k - x_{k+1})$$

即：

$$g_{k+1} - g_k = H_{k+1} \cdot (x_{k+1} - x_k)$$

拟牛顿条件

另一种推导方法是，直接利用微分中值定理，直接写出：

$$\nabla f(x_k) - \nabla f(x_{k+1}) = H_{k+1} \cdot (x_k - x_{k+1})$$

也得到：

$$g_{k+1} - g_k = H_{k+1} \cdot (x_{k+1} - x_k)$$

为简化表示，设 $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$ ，则得到拟牛顿条件：

$$y_k = H_{k+1} \cdot s_k \quad \text{即：} s_k = H_{k+1}^{-1} \cdot y_k$$

设 B_{k+1} 为 Hessian 矩阵 H_{k+1} 的近似，设 \hat{H}_{k+1} 为 H_{k+1}^{-1} 的近似矩阵，则：得到拟牛顿条件的惯用表示：

$$y_k = B_{k+1} \cdot s_k \quad \text{即：} s_k = \hat{H}_{k+1} \cdot y_k$$

拟牛顿法

观察拟牛顿条件：

$$y_k = B_{k+1} \cdot s_k \quad \text{即} : s_k = \hat{H}_{k+1} \cdot y_k$$

可知，在 y_k 与 s_k 已知的条件下：

- 若保持 B_{k+1} 或 \hat{H}_{k+1} 为对称矩阵，则其中有 $\frac{n^2+n}{2}$ 个未知数，而方程个数只有 k 个，所以满足条件的 B_{k+1} 或 \hat{H}_{k+1} 有无穷多个；所以，一定有多种满足拟牛顿条件的近似方法；
- 这些方法要么是对 B_{k+1} 进行迭代修正，要么是对 \hat{H}_{k+1} 进行迭代修正。

那么，如何选择构造 Hessian 矩阵的近似矩阵？

拟牛顿法之 DFP 方法

由数学家 William C. Davidon 在 1959 年最早提出，后经 Roger Fletcher、Michael J.D. Powell 在 1963 年完善，1991 年发表。

基本原理是，在前文分析的三个条件的限制下，对 \hat{H}_k 进行秩 2 修正：

$$\begin{aligned}\hat{H}_{k+1} &= \hat{H}_k + \Delta \hat{H}_k \\ &= \hat{H}_k + \alpha_k u_k u_k^T + \beta_k v_k v_k^T\end{aligned}$$

其中， $\alpha_k, \beta_k \in \mathcal{R}; u_k, v_k \in \mathcal{R}^n$ ；

由于 $u_k u_k^T$ 与 $v_k v_k^T$ 的秩均为 1，所以被称为“对 \hat{H}_k 进行秩 2 修正”；

接下来，关键看如何求解 $\alpha_k, \beta_k, u_k, v_k$ ：

拟牛顿法之 DFP 方法

DFP 秩 2 修正的好处：

- ① 在 u_k 与 v_k 不为零, α_k, β_k 不为负的前提下, 保证了 \hat{H}_{k+1} 矩阵的正定特性, 确保了对 x 稳定的调整方向;

以 u_k 为例： $z^T u_k u_k^T z = (u_k^T z)^T u_k^T z = \|u_k^T z\|^2 > 0; \Rightarrow$ 正定.

- ② $u_k u_k^T$ 与 $v_k v_k^T$ 均为对称矩阵, 从而延续了 \hat{H}_{k+1} 的对称性;
- ③ 当然, 接下来还得让它必须满足“拟牛顿条件”。

拟牛顿法之 DFP 方法

由拟牛顿条件：

$$s_k = \hat{H}_{k+1} \cdot y_k$$

代入 \hat{H}_{k+1} 的秩 2 修正，得：

$$\begin{aligned} s_k &= (\hat{H}_k + \alpha_k u_k u_k^T + \beta_k v_k v_k^T) y_k \\ &= \hat{H}_k y_k + \alpha_k u_k u_k^T y_k + \beta_k v_k v_k^T y_k \end{aligned}$$

即：

$$s_k - \hat{H}_k y_k = \alpha_k u_k (u_k^T y_k) + \beta_k v_k (v_k^T y_k)$$

也就是说，要满足拟牛顿条件，就是要使这个式子成立。

拟牛顿法之 DFP 方法

使这个式子成立的办法很多，DFP 方法中选择如下条件：

$$s_k = \alpha_k u_k (u_k^T y_k)$$

$$\hat{H}_k y_k = -\beta_k v_k (v_k^T y_k)$$

进而，可以再选择：

选择: $s_k = u_k$, 得到: $\alpha_k (u_k^T y_k) = 1$

选择: $\hat{H}_k y_k = v_k$, 得到: $\beta_k (v_k^T y_k) = -1$

进而得到：

$$u_k = s_k, \quad \alpha_k = \frac{1}{(u_k^T y_k)}$$

$$v_k = \hat{H}_k y_k, \quad \beta_k = -\frac{1}{(v_k^T y_k)} = -\frac{1}{(y_k^T \hat{H}_k y_k)}$$

从 DFP 到 BFGS 方法

最终得到，DFP 方法近似矩阵 \hat{H}_k 的迭代更新公式：

$$\hat{H}_{k+1} = \hat{H}_k + \frac{s_k^T s_k}{(u_k^T y_k)} - \frac{\hat{H}_k y_k y_k^T \hat{H}_k}{(y_k^T \hat{H}_k y_k)}$$

类似的，利用拟牛顿公式的另一个描述 $y_k = B_{k+1} \cdot s_k$ ，我们也可以对 B_{k+1} 进行迭代修正——BFGS 方法

- 以数学家 Broyden, Fletcher, Goldfarb, Shanno 的名字命名；
- 比 DFP 方法具有更好的性能，是当前常用的优化算法；

拟牛顿法之 BFGS 方法

对 B_{k+1} 进行秩 2 修正：

$$B_{k+1} = B_k + \alpha_k u_k u_k^T + \beta_k v_k v_k^T$$

由拟牛顿条件 $y_k = B_{k+1} \cdot s_k$, 得：

$$y_k = B_k s_k + \alpha_k u_k u_k^T s_k + \beta_k v_k v_k^T s_k$$

选取：

$$y_k = \alpha_k u_k u_k^T s_k = \alpha_k u_k (u_k^T s_k)$$

$$B_k s_k = -\beta_k v_k v_k^T s_k = -\beta_k v_k (v_k^T s_k)$$

再选取：

$$u_k = y_k, \text{ 得到： } \alpha_k = \frac{1}{u_k^T s_k}$$

$$v_k = B_k s_k, \text{ 得到： } -\beta_k = -\frac{1}{v_k^T s_k} = -\frac{1}{s_k^T B_k s_k}$$

拟牛顿法之 BFGS 方法

从而得到，BFGS 方法近似矩阵 B_k 的迭代更新公式：

$$B_{k+1} = B_k + \frac{y_k y_k^T}{u_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$$

注意： B_{k+1} 是对 H_{k+1} 的近似，在牛顿法中计算 $d_{k+1} = -H_{k+1}^{-1} \cdot g_{k+1}$ 的公式相应变为 $d_{k+1} = -B_{k+1}^{-1} \cdot g_{k+1}$ ，可见，还需要计算 B_{k+1}^{-1} 。

这很不爽，因此，改写公式：

$$B_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) B_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}$$

可见，BFGS 方法具有更好的计算性能，其时间复杂度为 $O(n^2)$ 级；然而，其空间复杂度仍值得关注： B_{k+1} 为 $N \times N$ 方阵，存储量很大；

拟牛顿法之 L-BFGS 方法

L-BFGS (Limited-storage BFGS) 基本思想：

- ① 在计算过程中，不选择存储 B_k ，而选择利用 $\{s_i\}\{y_i\}$ 的历史序列计算得到；
- ② 计算中，抛弃 m 步迭代之前的 $\{s_i\}\{y_i\}$ 序列，而仅利用最近的 m 个存储序列来估算；

对照公式：

$$B_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) B_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}$$

设： $\rho_k = \frac{1}{y_k^T s_k}$, $U_k = I - \rho_k y_k s_k^T$, $M_k = B_k^{-1}$ ，则原公式写为：

$$M_{k+1} = U_k^T M_k U_k + \rho_k s_k s_k^T$$

可见，计算 M_{k+1} 只需要对第一项进行 m 步迭代计算；

Thanks.