梯度下降法

王璐

梯度下降法 (Gradient Descent) 也称最速下降法 (Steepest Descent), 是常用的求目标函数极小值的一阶算法。这里的 “一阶” 是指在优化过程中只用到目标函数和其一阶导数（梯度）的信息，没有用到更高阶导数的信息。当目标函数 $f(\boldsymbol{x})$ 一阶可导时，由于函数在任意一点的负梯度方向是函数下降最快的方向，我们总可以沿函数负梯度 $-\nabla f(\boldsymbol{x})$ 的方向搜索，使得每一步的目标函数值都在减小。

与二阶优化算法 (e.g. Newton-Raphson) 相比，一阶算法通常收敛很慢。但对于变量个数很多的高维优化问题，一阶算法可能更有优势，因为每步迭代计算简单，而求二阶导数 (Hessian matrix) 的成本可能太高或不可行。特别当目标函数不是凸函数 (convex) 时，有时找到一个局部极小值点就足够了，而 Newton-Raphson 找到的可能是 “鞍点” (saddle point).

1 Gradient Descent (GD)

考虑下面的优化问题

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

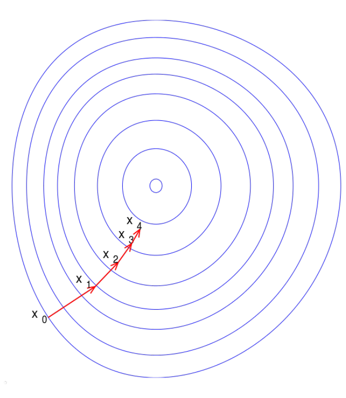
其中 $\boldsymbol{x} \in \mathbb{R}^d$, f 一阶可导。对于 $\mathbb{R}^d$ 上的任意单位向量 $\boldsymbol{v}$, f 沿 $\boldsymbol{v}$ 方向的改变率（方向导数）

$$\nabla_{\boldsymbol{v}} f(\boldsymbol{x}) = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x})}{h} = \nabla f(\boldsymbol{x})^\top \cdot \boldsymbol{v} = \|\nabla f(\boldsymbol{x})\| \cos(\theta)$$

其中 θ 是 $\boldsymbol{v}$ 与梯度 $\nabla f(\boldsymbol{x})$ 的夹角，当 $\boldsymbol{v}$ 与负梯度 $-\nabla f(\boldsymbol{x})$ 方向相同时，f 下降得最快。

- **Gradient Descent**

  1. 选取初始值 $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$
  2. 重复以下迭代直到收敛:
     - 计算 $\boldsymbol{g}_{t-1} = \nabla f(\boldsymbol{x}^{(t-1)})$
     - 选取步长 λ_t
     - 令 $\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \lambda_t \boldsymbol{g}_{t-1}$



在上述算法中，“迭代收敛” 可以是梯度模长小于某个临界值 $\left\|\nabla f(\boldsymbol{x}^{(t)})\right\| < \varepsilon$, 或两次迭代间函数变化小于某个临界值 $\left\|f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^{(t)})\right\| < \varepsilon$.

如果步长 λ_t 足够小，目标函数值会随着迭代进行不断减小，$f(\boldsymbol{x}^{(0)}) \geq f(\boldsymbol{x}^{(1)}) \geq f(\boldsymbol{x}^{(2)}) \geq \cdots$. 如果函数 f 存在有界的下限，$f(\boldsymbol{x}) \geq f^\star, \forall \boldsymbol{x}$, 那么序列 $\{\boldsymbol{x}^{(t)}\}$ 会收敛到 f 的一个局部极小值点。

1.1 步长 λ_t 的选取

如果选取固定步长，比如令 $\lambda_t = \lambda, t = 1, 2, \ldots$，当 λ 很大时，GD 算法可能会发散，如图1的左图所示；如果 λ 很小，GD 会收敛得很慢，如图1中间的图所示。通过后面的收敛性分析，我们可能会找到一个较合适的步长，如图1的右图所示。

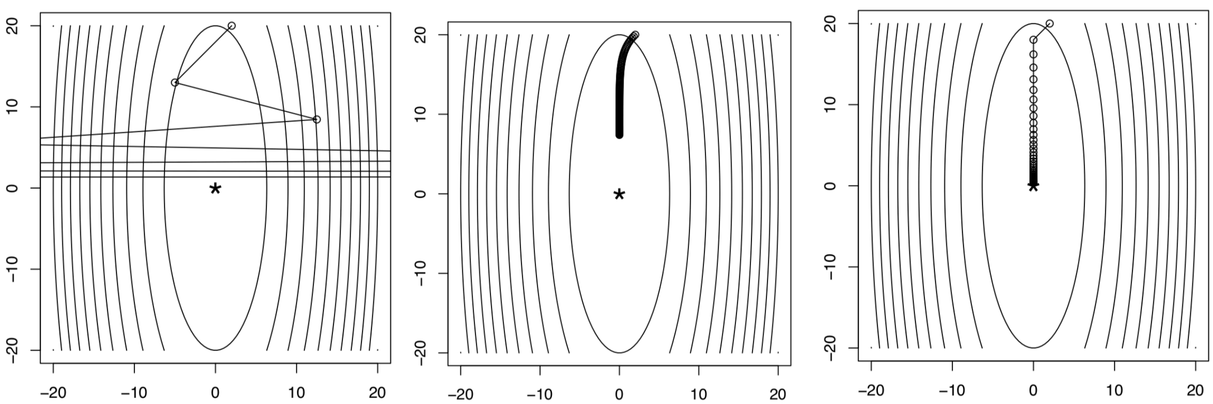


Figure 1: Gradient descent 中步长的选取对算法收敛的影响。Picture source: Ryan Tibshirani

常用的做法是让 λ_t 随迭代变化，每一步根据当前函数值和梯度选取合适的步长。Backtracking line search（一维搜索）是一种常用的自适应 (adaptively) 选取步长的方法。

- **Backtracking line search**

  1. 选取两个固定参数 $0 < \beta < 1$ 和 $0 < \alpha \leq 1/2$.
  2. 在第 t 步迭代开始时，选取一个较大的初始步长 γ_0, 然后不断缩小步长，令 $\gamma_j = \beta\gamma_{j-1}, j = 1, 2, \ldots$, 直到
  $$f\left(\boldsymbol{x}^{(t-1)} - \gamma_j \boldsymbol{g}_{t-1}\right) \leq f(\boldsymbol{x}^{(t-1)}) - \alpha\gamma_j \left\|\boldsymbol{g}_{t-1}\right\|_2^2. \tag{1}$$
  (1)被称为 Armijo 条件。
  3. 将满足 Armijo 条件(1)的步长 γ_j 选为第 t 步步长 λ_t.

在图1的例子中使用 backtracking line search 选取步长，$\alpha = \beta = 0.5$, 迭代的轨迹如图2所示。

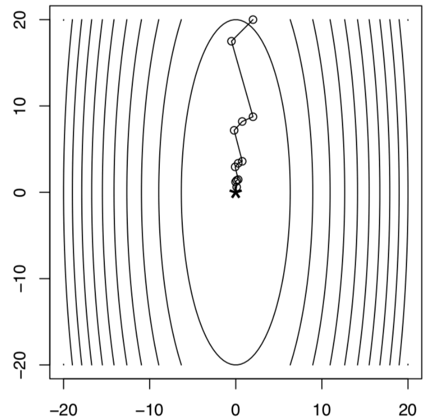


Figure 2: 12 步外循环，总共 40 步迭代。Picture source: Ryan Tibshirani

Remarks

1. 为了简化一维搜索的调参，一般令 $\beta = 1/2$.
2. 有时人们会选取非常小的 α, e.g. $\alpha = 10^{-4}$, 原因是宁可让目标函数下降得少一点，也不想尝试太多个 γ_j.
3. 更加复杂的一维搜索方法还有 Wolfe condition (Nocedal and Wright, 2006), 可以确保 α 不会太小。
4. 寻找最优步长的 exact line search:

$$\lambda_t = \underset{\lambda > 0}{\operatorname{argmin}}\ f(\boldsymbol{x}^{(t-1)} - \lambda \boldsymbol{g}_{t-1})$$

   在实践中一般不可行，因为计算成本太高。

1.2 收敛性分析

本节我们想回答下面两个问题：

- 步长 λ_t 对 GD 收敛的影响?
- 当 GD 收敛时，$\nabla f(\boldsymbol{x}^{(t)}) \to 0$, $t \to \infty$. 但可能对任何有限的 t, $\nabla f(\boldsymbol{x}^{(t)})$ 不会严格为 $\mathbf{0}$. 那么对任意一个给定的 $\varepsilon > 0$, 需要迭代多少步才能达到 $\left\|\nabla f(\boldsymbol{x}^{(t)})\right\| < \varepsilon$?

为了回答以上问题，我们需要假设

1. f 存在有界的下限 $f^\star$.

2. f 的梯度是 Lipschitz 连续。

Definition 1.1. 如果存在一个常数 $L > 0$, 使得对任意的 $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ 都有

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\|_2 \leq L \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$$

称 f 的梯度 $\nabla f(\boldsymbol{x})$ 是 **Lipschitz 连续**。

Remarks

- 梯度 Lipschitz 连续要求函数 f 的梯度不能 “变化得太快”。
- 梯度 Lipschitz 连续是一个很弱的假设，很多统计模型的 log-likelihood 都满足该条件，比如线性回归，logistic regression.
- 对于 C^2 函数 (二阶连续可导) f, 梯度的 Lipschitz 连续意味着 f 的 Hessian matrix $\nabla^2 f(\boldsymbol{x}) \preceq LI, \forall \boldsymbol{x}$. 即 $LI - \nabla^2 f(\boldsymbol{x})$ 总是一个半正定矩阵：$\forall \boldsymbol{x}$, $\boldsymbol{h}$,
$$\boldsymbol{h}^\top \nabla^2 f(\boldsymbol{x}) \boldsymbol{h} \leq L \|\boldsymbol{h}\|_2^2 .$$
- 对于 C^2 函数 f, 可将 L 选为 f 的 Hessian matrix 特征值的一个上界。比如在线性回归中，log-likelihood 的 Hessian 是 $X^\top X$, 因此 L 最小可取为 $X^\top X$ 的最大特征值。

对于 C^2 函数 f, 将 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}^{(t)}$ 处 Taylor 展开：

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)}) + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}^{(t)})^\top \nabla^2 f(\tilde{\boldsymbol{x}}) (\boldsymbol{x} - \boldsymbol{x}^{(t)}). \tag{2}$$

由 ∇f 的 Lipschitz 连续得

$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x} - \boldsymbol{x}^{(t)}) + \frac{L}{2} \left\|\boldsymbol{x} - \boldsymbol{x}^{(t)}\right\|_2^2 . \tag{3}$$

(3)的右端是 f 的一个二次上界函数且经过点 $(\boldsymbol{x}^{(t)}, f(\boldsymbol{x}^{(t)}))$，如图3所示。

将 GD 迭代公式 $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \lambda_t \boldsymbol{g}_t$ 代入(3)，其中 $\boldsymbol{g}_t = \nabla f(\boldsymbol{x}^{(t)})$，得到

$$f(\boldsymbol{x}^{(t+1)}) \leq f(\boldsymbol{x}^{(t)}) - \|\boldsymbol{g}_t\|_2^2 \lambda_t + \frac{L}{2} \|\boldsymbol{g}_t\|_2^2 \lambda_t^2 . \tag{4}$$

因此选取步长

$$\lambda_t = 1/L$$

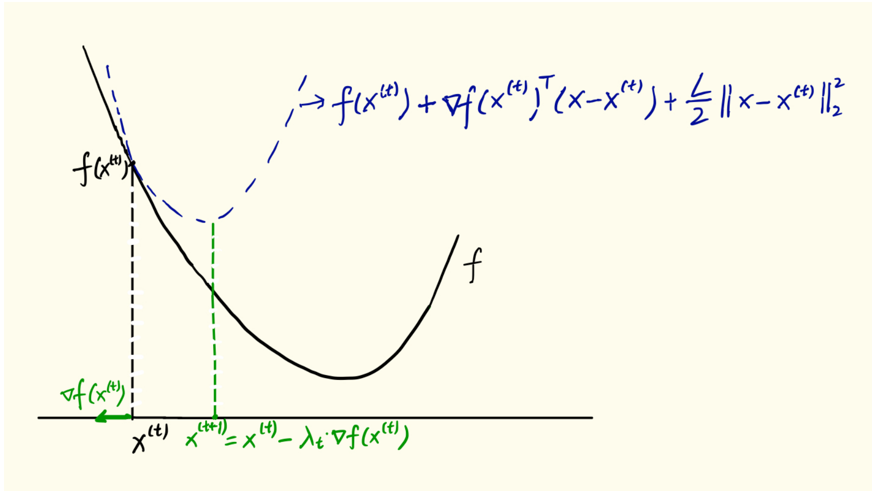


Figure 3: Gradient descent 最优步长的选取。

可使 $f(\boldsymbol{x}^{(t+1)})$ 的上界(4)达到最小。考虑在 GD 中选取常数步长 $\lambda_t \equiv 1/L$，由(4)得 $\forall t$,

$$f(\boldsymbol{x}^{(t+1)}) \leq f(\boldsymbol{x}^{(t)}) - \frac{1}{2L} \|\boldsymbol{g}_t\|_2^2 . \tag{5}$$

(5)说明，只要每步迭代中的梯度 $\boldsymbol{g}_t$ 不为 $\mathbf{0}$, 选取步长 $\lambda_t = 1/L$ 可以保证目标函数一直在下降，且每步迭代中目标函数减小的值与当前函数梯度的大小有关。

Remarks

1. 由(4)可得

$$f(\boldsymbol{x}^{(t+1)}) \leq f(\boldsymbol{x}^{(t)}) - \lambda_t (1 - \frac{L}{2}\lambda_t) \|\boldsymbol{g}_t\|_2^2 .$$

因此只要选取的步长足够小 ($\lambda_t < 2/L$)，就能保证 GD 算法收敛。这也说明在实践中选取过大的步长可能导致 GD 发散。

2. 虽然为了证明方便，我们假设 $f \in C^2$. 但对 C^1 函数，不等式(3)也成立。

$$\begin{aligned}
f(\boldsymbol{x}^{(t+1)}) &= f(\boldsymbol{x}^{(t)}) + \int_0^1 \nabla f\left(\boldsymbol{x}^{(t)} + \alpha(\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)})\right)^\top (\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}) d\alpha \qquad (6)\\
&= f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}) + \int_0^1 \left[\nabla f\left(\boldsymbol{x}^{(t)} + \alpha(\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)})\right) - \nabla f(\boldsymbol{x}^{(t)})\right]^\top (\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}) d\alpha \\
&\leq f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}) + \int_0^1 \left\|\nabla f\left(\boldsymbol{x}^{(t)} + \alpha(\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)})\right) - \nabla f(\boldsymbol{x}^{(t)})\right\| \cdot \left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\right\| d\alpha \\
&\leq f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}) + \int_0^1 L\alpha \left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\right\|^2 d\alpha \qquad (7)\\
&= f(\boldsymbol{x}^{(t)}) + \nabla f(\boldsymbol{x}^{(t)})^\top (\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}) + \frac{1}{2} L \left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\right\|^2
\end{aligned}$$

其中(6)根据微积分基本定理，(7)根据 Lipschitz 连续。

下面计算将误差降到 ε 以下所需 GD 迭代的步数。

为了简化问题，我们将步长固定在 $\lambda_t = 1/L$. (5)表明，从某个初始值 $f(\boldsymbol{x}^{(0)})$ 开始，GD 在每一步迭代中都会将 f 减小 $\frac{1}{2L}\|\boldsymbol{g}_t\|_2^2$. 对 (5)重新整理可得

$$\|\boldsymbol{g}_t\|_2^2 \leq 2L\left(f(\boldsymbol{x}^{(t)}) - f(\boldsymbol{x}^{(t+1)})\right)$$

由此得到:

$$\begin{aligned}
\|\boldsymbol{g}_{t-1}\|_2^2 &\leq 2L\left(f(\boldsymbol{x}^{(t-1)}) - f(\boldsymbol{x}^{(t)})\right)\\
&\vdots\\
\|\boldsymbol{g}_0\|_2^2 &\leq 2L\left(f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^{(1)})\right)
\end{aligned}$$

将以上不等式相加得到

$$\sum_{k=0}^{t-1}\|\boldsymbol{g}_k\|_2^2 \leq 2L\left(f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^{(t)})\right)$$

由于 $f(\boldsymbol{x}^{(t)}) \geq f^\star, \forall t$,

$$t \cdot \min_k \|\boldsymbol{g}_k\|_2^2 \leq \sum_{k=0}^{t-1}\|\boldsymbol{g}_k\|_2^2 \leq 2L\left(f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^{(t)})\right) \leq 2L\left(f(\boldsymbol{x}^{(0)}) - f^\star\right)$$

最终得到

$$\min_k \|\boldsymbol{g}_k\|_2^2 \leq \frac{2L\left(f(\boldsymbol{x}^{(0)}) - f^\star\right)}{t} = O\left(\frac{1}{t}\right). \tag{8}$$

(8)表明 GD 迭代 t 步后，至少在某一步 $k \in \{1,\ldots,t\}$ 有 $\|\boldsymbol{g}_k\|_2^2 = \left\|\nabla f(\boldsymbol{x}^{(k)})\right\|_2^2 = O(1/t)$. 即第 t 步迭代的误差是 $O(1/t)$, 也称 GD 的收敛速率 (convergence rate) 是 $O(1/t)$.

为了将梯度模长的平方降到 ε 以下，令

$$\frac{2L\left(f(\boldsymbol{x}^{(0)}) - f^\star\right)}{t} < \varepsilon$$

得到

$$t > \frac{2L\left(f(\boldsymbol{x}^{(0)}) - f^\star\right)}{\varepsilon}$$

所以 GD（最多）需要 $t = O(1/\varepsilon)$ 步迭代使得 $\left\|\nabla f(\boldsymbol{x}^{(k)})\right\|_2^2 < \varepsilon$.

像(8)这样，误差 ε 以 $O(1/t)$ 或 $O(1/t^2)$ 的速率减小的情况，被称为 **sublinear convergence**. 如果误差以 $O\left((1-\delta)^t\right)$ $(0 < \delta < 1)$ 的速率减小，称为 **linear convergence**. 这样命名的原因是后者在 $\log(\varepsilon)$ vs t 的图中看起来是线性下降的, 如图4的右图所示。如果实践中只需要一个低精度的解，sublinear rate 的算法可能就足够了。

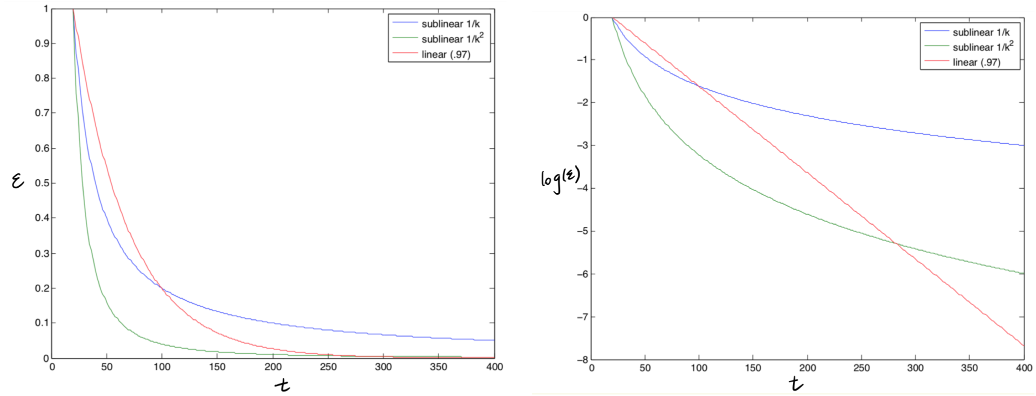


Figure 4: 不同收敛速度的算法迭代误差 ε 与迭代步数 t 的关系 (左)，$\log(\varepsilon)$ 与迭代步数 t 的关系 (右)。Picture source: Stephen Wright

---

总结

1. 以上我们证明了当目标函数的梯度是 Lipschitz 连续时，选取足够小的的步长可以保证 GD 收敛。
2. t 步迭代的误差是 $O(1/t)$. 这意味着要达到 $\left\|\nabla f(\boldsymbol{x}^{(k)})\right\|_2^2 < \varepsilon$，需要 $t = O(1/\varepsilon)$ 步迭代。

Remarks

1. 虽然在以上理论分析中，我们选取的步长是 $\lambda_t \equiv 1/L$, 但实践中一般不使用该步长，因为
   - L 通常很难计算
   - 即使可以找到满足条件的 L，$1/L$ 一般特别小，会导致 GD 收敛得很慢，只有在最坏的情况下、为保证收敛才使用。
2. Backtracking line search 是更实用的选取步长的方法，与(5)相比，很多情况下(1)中的 $\alpha\gamma_j > 1/(2L)$, 这样每一次迭代取得的进步比选取步长 $1/L$ 大。
3. GD 不适用于目标函数不可导的优化问题。

2 随机梯度下降 (Stochastic Gradient Descent)

统计模型中的目标函数通常是 joint log-likelihood, 且可以写成单个 log-likelihood 和的形式，即 $l(\theta \mid y_1, \ldots, y_n) = \sum_{i=1}^n l(\theta \mid y_i)$. 考虑如下优化问题

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^n f_i(\boldsymbol{\theta})$$

由于 $\nabla \sum_{i=1}^n f_i(\boldsymbol{\theta}) = \sum_{i=1}^n \nabla f_i(\boldsymbol{\theta})$，GD 迭代格式如下：

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \lambda_t \sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}^{(t)}),\ t = 0, 1, \ldots$$

随机梯度下降 (SGD) 的迭代格式为：

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \lambda_t \nabla f_{i_t}(\boldsymbol{\theta}^{(t)}),\ t = 0, 1, \ldots$$

其中 $i_t \in \{1, \ldots, n\}$.

有两种选择 i_t 的方式：

1. 循环式：令 $i_t = 1, 2, \ldots, n, 1, 2, \ldots, n, \ldots$.
2. 随机式：在每步迭代 t 随机选取 $i_t \in \{1, \ldots, n\}$.

实践中常用的是随机式。从计算量角度，n 步 SGD 迭代 ≈ 1 步 GD 迭代。它们在迭代精度上有什么区别？为了简化分析，我们采用固定步长和循环式 SGD.

- n 步 SGD 迭代：$\boldsymbol{\theta}^{(t+n)} = \boldsymbol{\theta}^{(t)} - \lambda \sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}^{(t+i-1)})$
- 1 步 GD 迭代：$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \lambda \sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}^{(t)})$

二者在方向上相差

$$\sum_{i=1}^n \left[\nabla f_i(\boldsymbol{\theta}^{(t+i-1)}) - \nabla f_i(\boldsymbol{\theta}^{(t)})\right].$$

如果每个 f_i 的梯度 $\nabla f_i(\boldsymbol{\theta})$ 不会随着 $\boldsymbol{\theta}$ 发生剧烈改变，则 SGD 与 GD 在更新方向上应该相差不大，也会收敛。

References

Nocedal, J. and Wright, S. (2006). *Numerical optimization.* Springer Science & Business Media.

