Line-Search Steepest Gradient Descent

Steepest Gradient Descent

$x^{k+1}=x^k-\tau\nabla f(x^k)$, where $\nabla f(x^k)$ is the grad or least-norm sub-grad of f, τ is the step size.

不同 τ 的选取方法 ($d=-\nabla f(x^k)$)

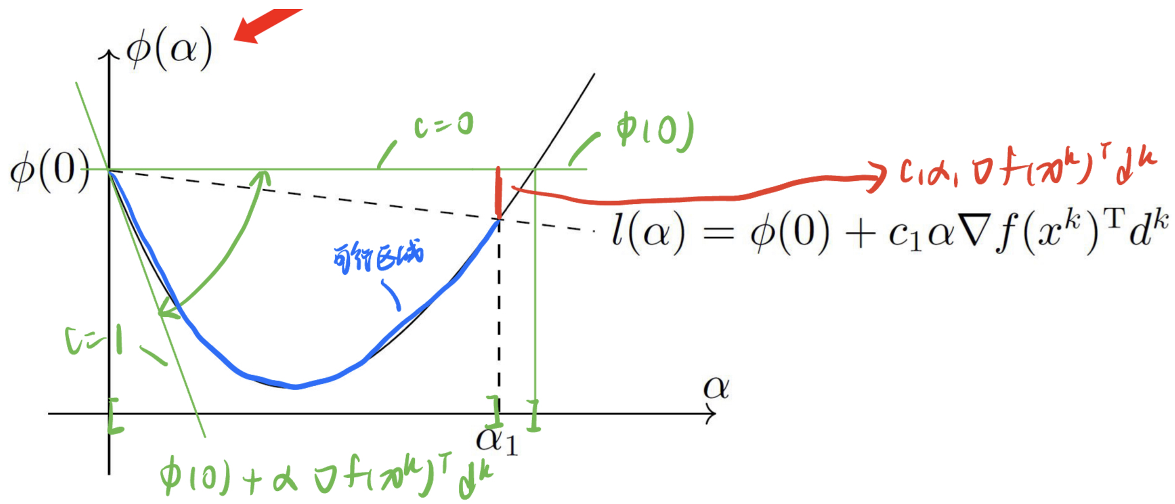
- constant $\tau=c$: 需要通过先验信息来合适地设置，过大会产生震荡，过小会使得收敛速度慢
- Diminishing $\tau=c/k$: 稳定性好且一定能收敛到局部极小，但收敛速度慢
- Exact line search $\tau=\arg\min_\alpha f(x^k+\alpha d)$: 需要的迭代次数少，但每一次迭代都需要求解一个最小化问题
- Inexact line search $\tau\in\{\alpha|f(x^k)-f(x^k+\alpha d)\geq -c\cdot\alpha d^T\nabla f(x^k)\}$

Inexact Line-Search Steepest Gradient Descent

适用于连续且分片光滑的函数。

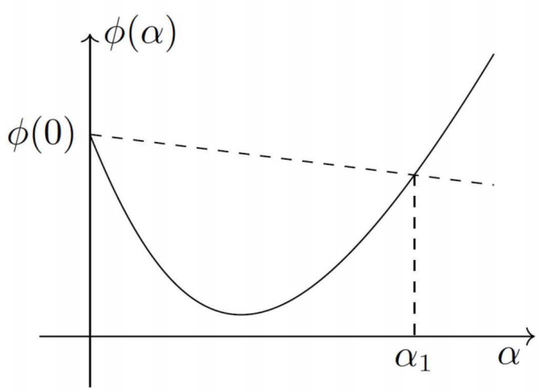
对于待优化的函数 $f(x^k)$ ，令 $\phi(\alpha)=f(x^k+\alpha d)$，则 $\phi(0)=f(x^k)$ ，函数在0处的斜率为 $\frac{\partial\phi(\alpha)}{\partial\alpha}|_{\alpha=0}=d^T\nabla f(x^k)$ ，令 $l(\alpha)=\phi(0)+c\cdot\alpha d^T\nabla f(x^k), c\in(0,1)$ ， c 取0和1时， $l(\alpha)$ 如图中绿色所示。

$\tau\in\{\alpha|f(x^k)+c\cdot\alpha d^T\nabla f(x^k)\geq f(x^k+\alpha d)\}, c\in(0,1)$ 表示，当 c 取 c_1 时，选取的步长 α 要使得 $f(x^k+\alpha d)$ 在直线 $l(\alpha)$ 的下方，图中蓝色部分为满足的区域，$f(x^k)+c\cdot\alpha d^T\nabla f(x^k)\geq f(x^k+\alpha d)$ 称为Armijo condition。



所以 τ 的可行域为 $[0,\alpha_1]$ 。此外，若 τ 选得过小会导致收敛过慢或无法收敛到局部极小。实际使用时令 $\tau=\tau_0$ ， τ_0 可以取得较大，然后不断令 $\tau\leftarrow\tau/2$ 直到 τ 满足Armijo condition。

Backtracking/Armijo line search



Choose search direction: $d = -\nabla f(x^k)$

While $f(x^k + \tau d) > f(x^k) + c \cdot \tau d^T \nabla f(x^k)$

$\tau \leftarrow \tau/2$

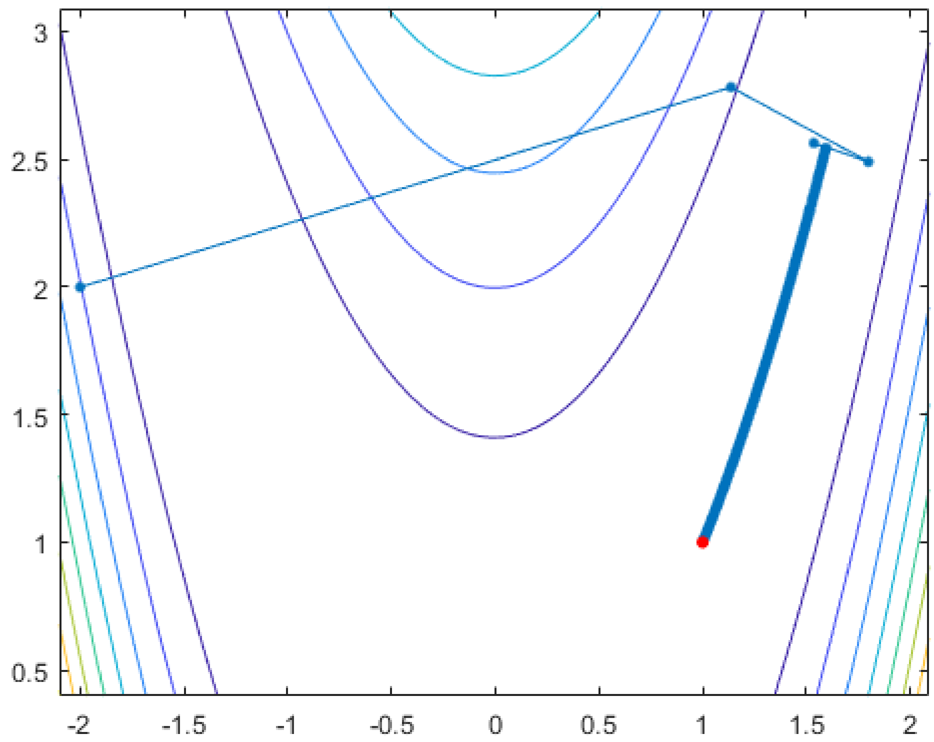
Update iterate $x^{k+1} = x^k + \tau d$

Repeat this until gradient is small
or subdifferential contains zero.

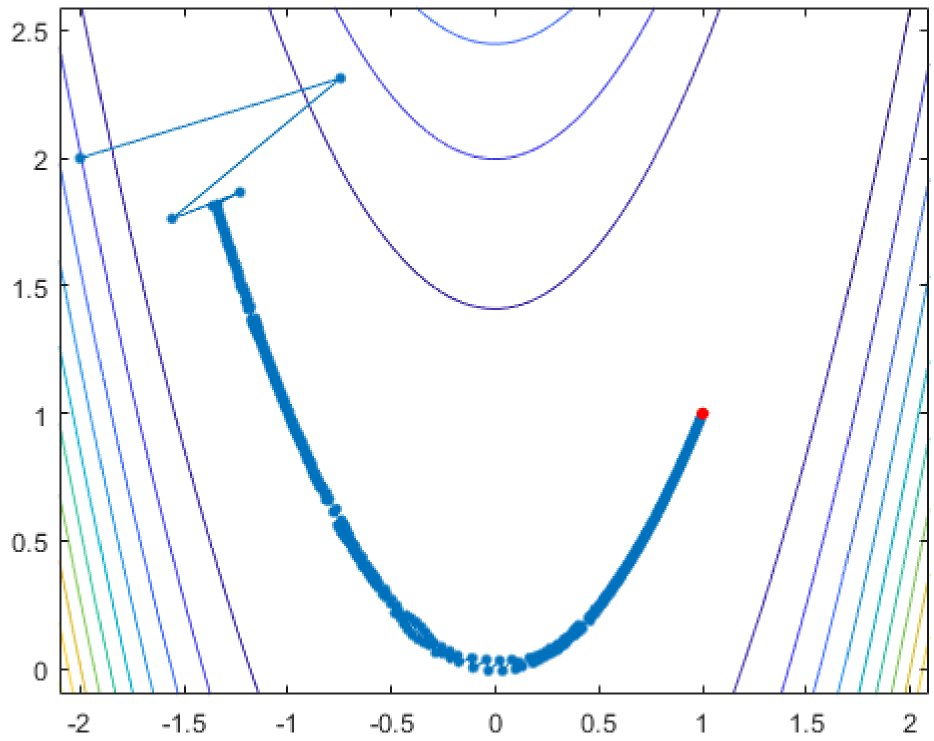
结果

2D

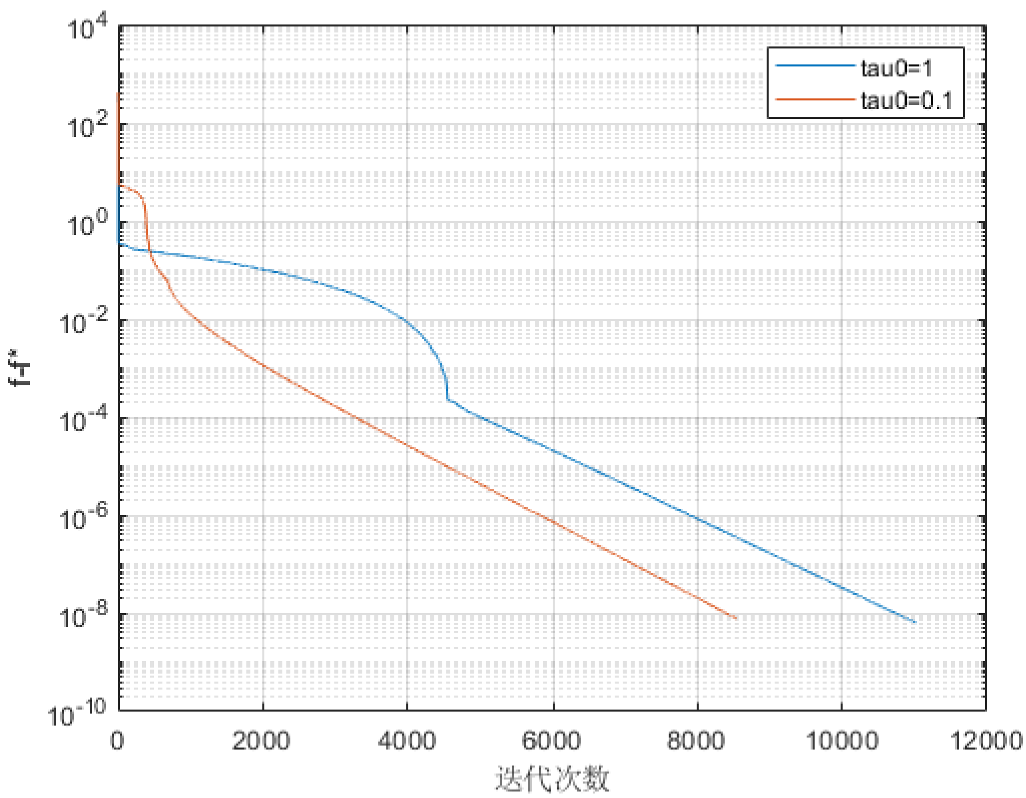
设置初解 $x_0 = [-2, 2]$，常数 $c = 10^{-3}$，初始步长 $\tau_0 = 1$，得到的结果如下图所示。从结果图可知，对于非凸的目标函数，Backtracking line search 算法适用于非凸目标函数的优化，可以越过局部最优，快速收敛到最优解，迭代次数为11021次，耗时1.038375s。



设置初始步长 $\tau_0 = 0.1$，其他数值不变，得到的结果如下图所示。当初始步长较小时，算法将当前解沿着函数减少的方向收敛到最优解，但迭代次数更少，只有8562次，耗时0.543456s。



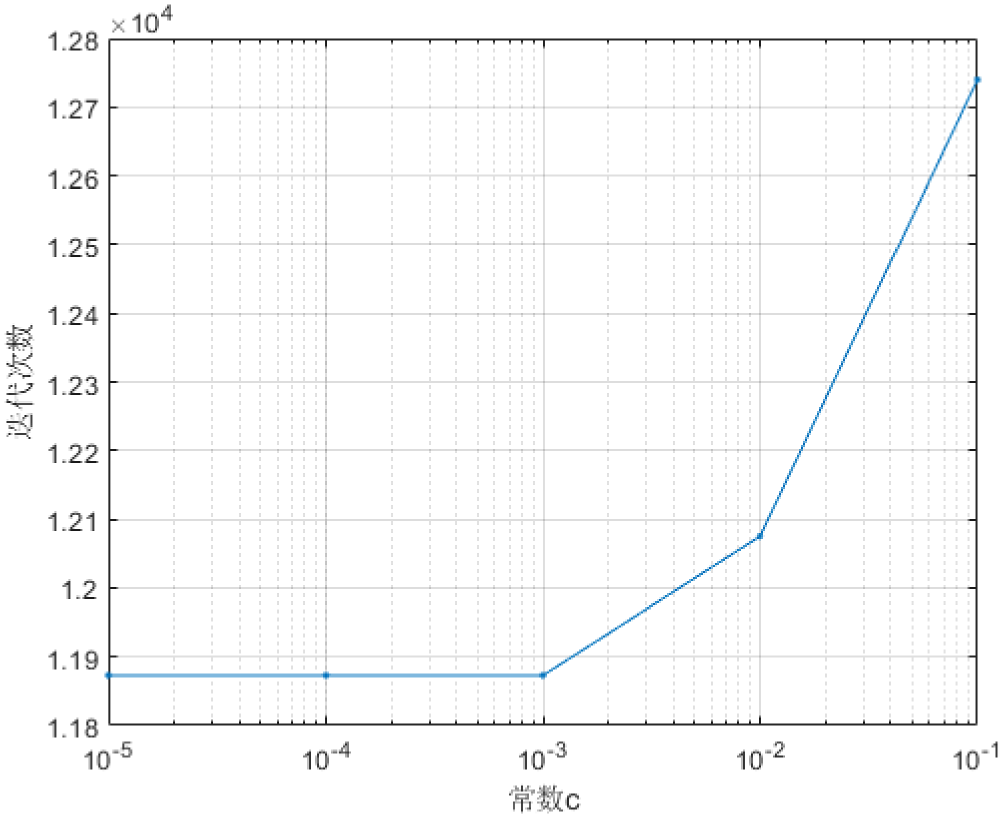
从迭代次数-函数值图可知，当初始步长为1时，在迭代的一开始函数值迅速下降，随后下降变得缓慢；而当步长为0.1时，函数值在迭代次数较小时都能快速减少，从而使得算法更快得找到最优解。导致这一结果的原因和函数的形式以及初解有关。



影响算法求解速度的因素研究

除了初解 x_0 外，算法还拥有两个参数，即常数 c 和初始步长 τ 。

首先是常数 c 对算法迭代次数的影响。分别设置常数 c 为 $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ ，得到算法的迭代次数如下图所示，从结果可知，常数 c 越小，算法的迭代次数也越少。



接着是初始步长 τ 对迭代次数的影响，结果如下图所示。由图可知，初始步长对于优化该函数所需的迭代次数的影响没有明显规律。