Line Search

Wenping Guo

1. 基本原理

线搜索是在搜索方向确定的前提下, 将梯度向量 g_k 投影到搜索方向 d_k 上, 进行一维搜索, 用于确定合理的步长 α_k.

位置坐标迭代公式:

$$x_{k+1} = x_k + \alpha_k d_k$$

其中 d_k 为搜索方向矢量(search direction).

定义如下一维函数:

$$\phi(\alpha) = f(x_k + \alpha d_k),\ \alpha > 0$$

对应的:

$\phi(0) = f(x_k), \phi(1) = f(x_k + d_k)$

将 function gradient g_k 投影到搜索方向矢量 d_k 上, 即得到线搜索的梯度, 其它为标量.

$$\phi'(0) = g_k^T d_k$$

$$\phi'(\alpha) = g_{k+1}^T d_k$$

使用二分法, 黄金分割法, 可以确定较为精确的 step size, 但计算量较大, 通常也没有必要. 在实际计算中, 应用的较多是近似搜索(inexact line search, approximate line search). line search 要控制步长别太大, 同时也不能太小. 一维搜索应返回较优的 α, 使得 ϕ 和 ϕ' 满足线搜索收敛性条件, 比如 Wolfe condition 等.

- 条件 1: The Sufficient Decrease Condition, 也称为 Armijo condition 确保能量降低足够大.

$$\mathrm{f}(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k g_k^T d_k$$

或写为:

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0)$$

- 条件 2: The Curvature Condition, 确保斜率变化足够大.

$$g_{k+1}^T d_k \geq c_2 g_k^T d_k$$

也可写为:

$$\phi'(\alpha) \geq c_2 \phi'(0)$$

- 条件 3: strong wolfe condition on curvature

$$\left|g_{k+1}^T d_k\right| \leq c_2 \left|g_k^T d_k\right|$$

或:

$$\left|\phi'(\alpha)\right| \leq c_2 \left|\phi'(0)\right|$$

满足条件 1 和条件 2, 称为 Wolfe conditions; 满足条件 1 和条件 3, 称为 Strong Wolfe conditions.

其中参数要求:

$$0 \leq c_1 \leq c_2 < 1$$

wikipedia 中提及, 通常情况下, c1 = 0.1, c2 = 0.4

几种重要的线搜索算法:

- Backtracking: 进退法, 最简单. 仅需要函数值.
- MoreThuente 算法, 需要计算梯度, 满足 strong wolfe conditions [1] 原 LBFGS 使用的就是这个算法. 以二次和三次插值为基础.
- HagerZhang [2]

2. 调用接口

使用 builder 自定义搜索参数, 使用 find 函数来优化步长. find 函数需要定义回调函数, 用于计算单变量函数 ϕ 的函数值 $\phi(\alpha)$ 及梯度 $\phi'(\alpha)$.

Listing 1:

```
Line search, also called one-dimensional search, refers to an optimization
procedure for univariable functions.

# Available algorithms

* MoreThuente
* BackTracking
* BackTrackingArmijo
* BackTrackingWolfe
* BackTrackingStrongWolfe

# References

* Sun, W.; Yuan, Y. Optimization Theory and Methods: Nonlinear Programming, 1st
  ed.; Springer, 2006.
* Nocedal, J.; Wright, S. Numerical Optimization; Springer Science & Business
```

```
  Media, 2006.
```

Listing 2:

```
use line::linesearch;

let mut step = 1.0;
let count = linesearch()
    .with_max_iterations(5) // the default is 10
    .with_initial_step(1.5) // the default is 1.0
    .with_algorithm("BackTracking") // the default is MoreThuente
    .find(|a: f64| {
        // restore position
        x.veccpy(&x_k);
        // update position with step along d
        x.vecadd(&d_k, a);
        // update value and gradient
        let phi_a = f(x, &mut gx)?;
        // update line search gradient
        let dphi = gx.vecdot(d);
        // update optimal step size
        step = a;
        // return the value and the gradient in tuple
        (phi_a, dphi)
    })?;
```

3. References

- Line search - Wikipedia
- Wolfe conditions - Wikipedia
- Backtracking line search - Wikipedia
- climin/linesearch.py at master · BRML/climin
- pysisyphus/BacktrackingOptimizer.py at dev · eljost/pysisyphus
- JuliaNLSolvers/LineSearches.jl: Line search methods for optimization and root-finding

(1) Moré, J. J.; Thuente, D. J. Line Search Algorithms with Guaranteed Sufficient Decrease. *ACM Trans. Math. Softw. TOMS* **1994**, *20* (3), 286–307.

(2) Hager, W. W.; Zhang, H. Algorithm 851: CG_DESCENT, a Conjugate Gradient Method with Guaranteed Descent. *ACM Trans. Math. Softw. TOMS* **2006**, *32* (1), 113–137.