Lecture 14: Line Search

Sondre Pedersen

Feb 21, 2025

1 Intro

The objective of line search: make the gradient algorithms work when you start far away from optimum. These algorithms are sometimes called globalization strategies. For line search we have two elements: (i) conditions on step-length, Wolfe conditions and (ii) Step-length computation.

Line search iterates: $x_{k+1} = x_k + \alpha_k p_k$. Gradient descent directions: $p_k = -B_k^{-1}\nabla f(x_k) = -B_k^{-1}\nabla f_k$.

- $B_k = I$: Steepest descent
- $B_k = \nabla^2 f_k$: Newton
- $B_k \approx \nabla^2 f_k$: Quasi-Newton

Observation: $B_k > 0 \implies p_k = -B_k^{-1}\nabla f_k$ descent direction.
Proof: $p_k^\top \nabla f_k = -\nabla f_k^\top B_k^{-1}\nabla f_k < 0$. ($B_k > 0 \implies B_k^{-1} > 0$).
Note: Hessian is not necessarily positive definite (far from solution).

Topics today - How to choose α_k and how to ensure $B_k > 0$.

2 Descent

Define $\phi(\alpha) = f(x_k + \alpha p_k)$. Here the input α determines how far we should walk along the direction p_k.
One strategy is to use exact linesearch. $\alpha^* = arg\min_\alpha \phi(\alpha)$. Not done in practice because α^* is too expensive to calculate, and not even necessary. Instead, do inexact linesearch. Find 'cheap' α that fulfills (i) sufficient decrease (Armijo), and (ii) Desired slope (curvature).

Why sufficient decrease? To make sure you actually make progress. Otherwise it is possible to keep going over the goal. This is the 1st Wolfe condition, or the Armijo condition. It says choose α that fulfills $\phi(\alpha) \leq \phi(0) + c_1\alpha\phi(0) := l(\alpha)$ This condition allows for very small steps.

2nd Wolfe condition: Desired slope. $\phi'(\alpha) > c_2\phi'(0),\ c_2 \in (c_1, 1)$ The rationale is that we should not stop when $\phi(\alpha) << 0$. Typical value Newton/Q-N is $c_2 = 0.9$.

Summary - Good step lengths should fulfill the Wolfe conditions:

- $f(x_k + \alpha p_k) \leq f(x_k) + c_1\alpha_k \nabla f_k^\top p_k$
- $\nabla f(x_k + \alpha p_k)^\top p_k \geq c_2 \nabla f_k^\top p_k$

How do we compute the step length then?

Backtracking Line Search - Algorithm 3.1

```
1. Choose ᾱ > 0, p ∈ (0,1), c ∈ (0,1); Set α ← ᾱ
2. repeat until f(x_k + αp_k) ≤ f(x_k) + cα∇f_k^⊤ p_k
3.  α ← pα
4. end (repeat)
5. Terminate with α_k = α
```

Very easy to implement. A problem is how to choose $\overline{\alpha}$? In Newton/Q-N it is easy; start with $\overline{\alpha} = 1$ and reduce. But in steepest descent there is no general way to pick a good value.

Interpolation -

1. (1) Initialize with α_0 (initial guess)
2. If α_0 fulfuills Wolfe -> exit
3. (2) Quadratic Interpolation
4. $Q_q(\alpha) = a\alpha^2 + \phi'(0)\alpha_\tau\phi(0)$
5. $a = (\phi(\alpha_0) - \phi(0) - \alpha_0\phi'(0))/(\alpha_0^2)$
6. $\alpha_1 = arg\min_\alpha \phi_q(\alpha)$
7. If α_1 fulfills Wolfe -> exit
8. (3) Do cubic Interpolation
9. $\phi_c(\alpha) = a\alpha^3 + b\alpha^2 + \phi'(0)\alpha + \phi(0)$
10. a, b can be found with expression from p. 58
11. $\alpha_2 = arg\min_\alpha \phi_c(\alpha)$
12. If α_2 fulfills Wolfe -> exit
13. (4) Start over with $\alpha_0 = \alpha_2$

Example - Line search for convex quadratic objective function.

Note: it does not really make sense to use line search for this problem, because it can be solved directly. This is for demonstration purposes only. The problem is

$$f(x) = \frac{1}{2}x^\top Gx + c^\top x,\ G > 0$$
$$x_k,\ p_k \text{ given}$$

We get

$$\begin{aligned}
\phi(\alpha) &= \frac{1}{2}(x_k + \alpha p_k)^\top G(x_k + \alpha p_k) + c^\top(x_k + \alpha p_k) \\
&= \frac{1}{2}p_k^\top Gp_k\alpha^2 + x_k^\top Gp_k\alpha + c^\top p_k\alpha + const \\
\phi'(\alpha) &= p_k^\top Gp_k\alpha + (x_k^\top G + c^\top)p_k = 0 \\
\implies \alpha^* &= -\frac{(Gx_n + c)^\top p_k}{p_k^\top Gp_k} \\
\text{If Newton: } p_k &= -[\nabla^2 f_k]^{-1}\nabla f_k = -G^{-1}(Gx + c) \\
\implies \alpha^* &= \frac{(Gx_k + c)^\top G^{-1}(Gx_k + c)}{(Gx_k + c)^\top G^{-1}GG^{-1}(Gx_k + c)} = 1
\end{aligned}$$

3 Newton: Hessian modification

$$x_{k+1} = x_k + \alpha_k p_k,\ p_k = -[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$$

In practice we don't compute the inverse. Instead, just solve $\nabla^2 f_k p_k = -\nabla f_k$. To solve such a linear system, it is best to use Cholesky factorization. However, far from the solution, $\nabla^2 f_k$ is typically not positive definite. Then we can also not use Cholesky (can use LDL).

To fix these issues, modify the Hessian and replace $\nabla^2 f_k$ with $\nabla^2 f_k + E_k > 0$. E_k can be constructed in several ways. One example is $E_k = \tau_k I,\ \tau_k \begin{cases} 0,\ \nabla^2 f_k > 0 \\ -\lambda_{\min}(\nabla^2 f_k) + \Delta,\ \text{otherwise} \end{cases}$