INTRODUCTION TO LINE SEARCH METHODS

David F. Gleich

February 25, 2026

Recall our short-hand notation:

$$\begin{aligned} \mathbf{x} &= \mathbf{x}^{(k)} \\ \mathbf{x}^+ &= \mathbf{x}^{(k+1)} \\ \mathbf{g} &= \mathbf{g}(\mathbf{x}^{(k)}) \\ \mathbf{g}^+ &= \mathbf{g}(\mathbf{x}^{(k+1)}) \\ \boldsymbol{H} &= \boldsymbol{H}(\mathbf{x}^{(k)}) \\ f_k &= f(\mathbf{x}^{(k)}) \\ f^+ &= f(\mathbf{x}^{(k+1)}) \\ f_{k+1} &= f(\mathbf{x}^{(k+1)}) \end{aligned}$$

Consider the unconstrained optimization problem:

$$\text{minimize} \quad f(\mathbf{x})$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.

At a point $\mathbf{x}^{(k)}$, a line search method proceeds by choosing

$$\text{a search direction } \mathbf{p} \qquad \text{and} \qquad \text{a step length } \alpha$$

such that

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{p}.$$

Both of these terms are used repeatedly in line-search methods.

Line search methods are a type of optimization algorithm that follows a template. We'll see five different methods to choose a search direction:

- gradient descent
- Newton's method
- quasi-Newton
- conjugate gradients
- limited memory quasi-Newton.

Each of these is called a different algorithm, but there are all instances of the template algorithm:

```
Given $\mathbf{x}_1$
While **not** yet at a minimizer
  Generate a search direction $\mathbf{p}_k$
  Find a step-length $\alpha$
```

In subsequent lectures, we'll prove that this template algorithm converges if your choice of search direction and step length obey the rules.

For instance, search directions used for a line-search method should be "promising." We make this precise by requiring the search direction $\mathbf{p}$ to be a *descent direction*.

DEFINITION 1 (descent direction) *Consider an optimization problem*

$$\text{minimize} \quad f(\mathbf{x})$$

and a point $\mathbf{y}$. *The direction* $\mathbf{p}$ *is called a descent direction at* $\mathbf{y}$ *if* $\mathbf{p}^T\mathbf{g}(\mathbf{y}) < 0$.

Descent directions are nice because we can always find an α such that $f(\mathbf{x}+\alpha\mathbf{p}) < f(\mathbf{x})$ as the following theorem shows. That is to say, there's always a sufficiently small step α such that we reduce the objective function and hopefully get closer to a minimizer.

THEOREM 2 *Let* $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *be continuously differentiable with gradient* $\mathbf{g}(\mathbf{x})$. *Consider a point* $\mathbf{y}$ *and a descent direction* $\mathbf{p}$. *Then there exists* $\alpha > 0$ *such that* $f(\mathbf{y} + \alpha\mathbf{p}) < f(\mathbf{y})$.

Proof The proof is just an application of Taylor's theorem:

$$f(\mathbf{y} + \gamma\mathbf{p}) = f(\mathbf{y}) + \gamma\mathbf{p}^T\mathbf{g}(\mathbf{y}) + O(\gamma^2).$$

If γ is sufficiently small such that $\gamma \gg \gamma^2$, then

$$f(\mathbf{y} + \gamma\mathbf{p}) = f(\mathbf{y}) + \gamma\mathbf{p}^T\mathbf{g}(\mathbf{y}) + O(\gamma^2) \le f(\mathbf{y})$$

because $\gamma\mathbf{p}^T\mathbf{g}(\mathbf{y})$ is negative by virtue of being a descent direction and $\gamma > 0$. The theorem follows with α as the value of γ that achieves this bound. ■

Generating descent directions is also easy.

THEOREM 3 *Let $\boldsymbol{B}$ be a symmetric, positive definite matrix. Consider the direction $\mathbf{p}$ that solves the linear system $\boldsymbol{B}\mathbf{p} = -\mathbf{g}(\mathbf{y})$. Then $\mathbf{p}$ is a descent direction.*

Proof It's an easy one! Give it a try. ∎

Unfortunately, picking a descent direction alone will not allow us to guarantee that the line search template converges. Let's see an example of why:

EXAMPLE 4 *Consider $f(x) = \frac{1}{2}x^2$ and $x_1 = 2$. The gradient is $g(x) = x$ and so we satisfy the criteria of a decent direction if $p = -1$. Suppose we pick $\alpha = 5/2 = 2 + 1/2$ as our step length, then $x_2 = -3/2$. At which point, $p = 1$ is a descent direction, and we can pick $\alpha = 2 + 1/4 = 9/4$. We've decreased the objective at both steps, however, if we continue this pattern: $\alpha_k = 2 + \frac{1}{2^k}$, $p_k = (-1)^k$, then we'll generate a sequence of iterates where $f(x_{k+1}) < f(x_k)$, but where x_k converges to ± 1, not the minimizer at zero.*

This motivates a set of what are called *line search conditions.*

GRADIENT DESCENT

In gradient descent, we choose $\mathbf{p} = -\mathbf{g}$. That is, the search direction is the negative gradient. Hence the algorithm "descends" the gradient. The choice of the gradient is optimal in the sense that:

$$\mathbf{p} = -\mathbf{g}/\|\mathbf{g}\| \text{ solves } \begin{array}{ll} \text{minimize} & f(\mathbf{x}+\mathbf{p}) \approx f(\mathbf{x}) + \mathbf{p}^T\mathbf{g} + O(1) \\ \text{subject to} & \|\mathbf{p}\| = 1. \end{array}$$

That is, it's the best direction to minimize a first order Taylor expansion of the objective.

NEWTON

We get the Newton search direction by solving:

$$\boldsymbol{H}\mathbf{p} = -\mathbf{g}$$

This direction solves for the minimizer of a second order Taylor expansion of the objective, assuming that the Hessian evaluated at $\mathbf{x}$ is positive definite:

$$\mathbf{p} = -\boldsymbol{H}^{-1}\mathbf{g} \text{ solves } \text{minimize} \quad f(\mathbf{x}+\mathbf{p}) \approx f(\mathbf{x}) + \mathbf{p}^T\mathbf{g} + \tfrac{1}{2}\mathbf{p}^T\boldsymbol{H}\mathbf{p}.$$

We'll be able to show that using the Newton search direction results in a quadratically convergent optimization procedure. This will be really fast!

QUASI-NEWTON

Quasi-Newton methods do not require exact knowledge of the Hessian. Instead, they build an approximation to the Hessian as the algorithm progresses. They are commonly used when solving problems where the Hessian is not available – which is usually because it is difficult or expensive to compute. If $\boldsymbol{B}$ is a semi-definite approximation to the Hessian, then a Quasi-Newton method will use the search direction $\boldsymbol{B}\mathbf{p} = -\mathbf{g}$.

CONJUGATE GRADIENTS

The conjugate gradients is a non-linear extension of the famous conjugate gradient algorithm for solving a linear system. We'll discuss it soon.

LIMITED-MEMORY QUASI-NEWTON

For large scale problems, keeping an approximation of the Hessian matrix in a quasi-Newton method requires $O(n^2)$ memory because the matrices are usually dense. A limited-memory Quasi-Newton method stores a low-rank approximation of the Hessian approximation.