One-Dimensional Minimization

Before addressing the question of how to solve an n-dimensional unconstrained minimization problem, we will consider the particular case of $n = 1$, usually referred to as *one-dimensional minimization* or *line search*:

$$\min_{x\in[a,b]} f(x),$$

where $[a, b]$ is some search interval. In the sequel, we will see that many unconstrained minimization algorithms will iterate over the selection of a descent direction $\mathbf{d}$, construct a one-dimensional function $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ by sectioning $f(\mathbf{x})$ along the line in direction $\mathbf{d}$ passing through the current point $\mathbf{x}$. Since $\mathbf{d}$ is a descent direction, we will usually ignore negative solutions, $\alpha < 0$ (i.e., typically $a = 0$). Because of this fact, a more appropriate name for "line search" would probably be "ray search", referring to the ray $\{\mathbf{x}+\alpha\mathbf{d} : \alpha \geq 0\}$.

In what follows, we present a few very commonly used line search algorithms. For the simplicity of the analysis, we will assume that the one-dimensional function $f(x)$ is *strictly unimodal* (that is, has one minimum) on $[a, b]$ (note that this does not necessarily mean that f is convex!). If this is not true (as often happens in real life), the discussed algorithms will return one of the local minima.

1 Bisection

If the function is $\mathcal{C}^1$ and unimodal on $[a, b]$ with $x^* \in [a, b]$ being the sought minimizer, the first-order derivative f' is negative on $[a, x^*)$, zero at x^*, and positive on $(x^*, b]$. Therefore, we can rewrite our problem as finding the *root of the derivative*,

$$\text{Find } x \in [a, b] \text{ such that } f'(x) = 0.$$

Starting with the initial search interval $[a, b]$, we calculate the midpoint $\tau = \dfrac{a+b}{2}$. If $f'(\tau) > 0$, we know for sure that $x^* \in [a, \tau)$; otherwise, if $f'(\tau) < 0$, we know for sure that $x^* \in (\tau, b]$. As the result, we can restrict the search interval to either $[a, \tau]$ or $[\tau, b]$ and repeat the process. This leads to the following simple iterative algorithm:

input : derivative $f'(x)$; search interval $[a, b]$; maximum number of iterations N; tolerance parameter ϵ
output: (approximate) root τ of f'
for $i = 1, 2, \ldots, N$ **do**
  set $\tau = \dfrac{a+b}{2}$
  if $|f'(\tau)| < \epsilon$ **then** stop
  if $f'(\tau) > 0$ **then** set $b = \tau$ **else** set $a = \tau$
end

Algorithm 1: Bisection line search

The algorithm is terminated either by exceeding the maximum number of iterations N, or when the derivative is ϵ-close to zero. Note that the bisection line search halves the search interval size at every iteration. After k iterations, the interval size will be $\Delta_k = 2^{-k}\Delta_0$, with $\Delta_0 = b - a$, meaning that $N \approx 20 - 30$ iterations are sufficient for any practical purpose.

2 Golden section

The bisection algorithm requires access to the first-order derivative of $f(x)$. In some problems, the calculation of the derivative can be very computationally expensive. In such cases, we would like to construct a line search algorithm based on zero-order information (i.e., the values of $f(x)$) only. It is easy to see that simply evaluating $f(a)$, $f(b)$ and $f(p)$ at some $p \in (a, b)$ does not tell sufficient information about the location of the minimum: we can construct f having the minimum either in $[a, p]$ or in $[p, b]$. However, if we evaluate f at an additional point $q \in (p, b)$, the comparison of $f(p)$ and $f(q)$ allows to restrict the search interval: if $f(p) < f(q)$, we do not know if $x^* \in [a, p]$ or $x^* \in [p, q]$, but we know for sure that $x^* \notin [q, b]$. Similarly, if $f(p) > f(q)$, we know for sure that $x^* \notin [a, p]$. This allows to restrict the search interval to either $[a, q]$ or $[p, b]$.

The only question that remains to be answered is how to choose p and q. The described procedure requires to calculate four value of the function f at a, p, q, b; when the procedure is repeated iteratively, $f(a)$ and $f(b)$ are inherited from the previous iteration. To further reduce complexity, we can select p and q consistently in such a way that either $f(p)$ or $f(q)$ will be inherited from the previous iteration, thus reducing the number of function evaluations per iteration to one.

In order to achieve this, let us denote the initial search interval length by $\Delta_0 = b - a$, and the length of the interval $[q, b]$ as $\tau\Delta_0 = b - q$. Since we don't have any particular reason to prefer the right side of the interval, by symmetry we will also assume the length of the interval $[a, p]$ to be $\tau\Delta_0 = p - a$. This gives

$$p = a + \tau\Delta_0$$

and

$$q = a + (1 - \tau)\Delta_0.$$

Assume without loss of generality that $f(p) < f(q)$ (from symmetry considerations, the same will hold for $f(p) > f(q)$), so that the interval is updated to $[a', b'] = [a, q]$ with the length $\Delta_1 = (1 - \tau)\Delta_0$. Then, the new internal point q' is

$$q' = a + (1 - \tau)\Delta_1 = a + (1 - \tau)^2\Delta_0.$$

We will require that $q' = p$, obtaining the equation

$$a + (1 - \tau)^2\Delta_0 = a + \tau\Delta_0.$$

Subtracting a and dividing by Δ_0 yields

$$(1 - \tau)^2 = \tau.$$

This quadratic equation is known since the ancient time, and its solution,

$$\tau = \frac{3 - \sqrt{5}}{2} =\approx 0.382,$$

is related to the quantity $\phi = \dfrac{1 + \sqrt{5}}{2} \approx 1.618$ known as the *golden ratio*.

This choice gives rise to the *golden section* line search algorithm summarized below:

input : function $f'(x)$; search interval $[a, b]$; maximum number of iterations N;
tolerance parameter δ
output: (approximate) minimizer x^* of f
$p \leftarrow a + \tau(b - a)$ and $q \leftarrow b - \tau(b - a)$
$f_a \leftarrow f(a)$, $f_b \leftarrow f(b)$, $f_p \leftarrow f(p)$, and $f_q \leftarrow f(q)$
for $i = 1, 2, \ldots, N$ **do**
  if $|b - a| < \delta$ **then** break
  if $f_p < f_q$ **then**
    $b \leftarrow q$ and $f_b \leftarrow f_q$
    $q \leftarrow p$ and $f_q \leftarrow f_p$
    $p \leftarrow a + \tau(b - a)$ and $f_p \leftarrow f(p)$
  end
  else
    $a \leftarrow p$ and $f_a \leftarrow f_p$
    $p \leftarrow q$ and $f_p \leftarrow f_q$
    $q \leftarrow b - \tau(b - a)$ and $f_q \leftarrow f(q)$
  end
end
return $x^* = \dfrac{a + b}{2}$

Algorithm 2: Golden section line search

Note how the function values are reused at each iteration, so only one value of either $f(p)$ or $f(q)$ is computed. Also note that we stop the algorithm by the size of the search interval instead of the absolute value of the derivative in order to avoid the evaluation of the latter.

3 Quadratic interpolation

In some problems, the objective function being minimized is regular (e.g., C^n) and it is advantageous to use this fact to speedup the line search. Interpolation-based line search algorithms take advantage of this regularity by fitting another (simpler) function, whose minimizer is usually given in a closed form. The simplest of such algorithms is based on quadratic interpolation.

Given three points a, b and $p \in (a, b)$, we can fit a quadratic function $h(x) = \alpha x^2 + \beta x + \gamma$ passing through them; for that, we demand

$$\begin{aligned} f(a) &= \alpha a^2 + \beta a + \gamma \\ f(p) &= \alpha p^2 + \beta p + \gamma \\ f(b) &= \alpha b^2 + \beta b + \gamma. \end{aligned}$$

We leave as an exercise the solution of this system of linear equations and the derivation of a closed-form expression for the minimizer of $h(x)$.

Exercise 1. *Derive a closed-form expression for the minimizer of the quadratic function $h(x)$ in terms of $f(a)$, $f(b)$, and $f(p)$.*

Note that in order for the parabola $h(x)$ to have a minimum, we need to make sure that $f(p) < f(a), f(b)$ – such a configuration of points is called a *V-combination* (as their location on the graph of f resembles the letter V). As we have seen in our discussion of the golden section, a V-combination can always be selected out of four distinct points.

The quadratic interpolation line search can be summarized as the following algorithm:

input : function $f(x)$ and its derivative $f'(x)$; search interval $[a, b]$; maximum number of iterations N; tolerance parameter ϵ
output: (approximate) minimizer x^* of f
start with $p \in [a, b]$ forming a V-combination
for $i = 1, 2, \ldots, N$ **do**
  calculate minimizer q of the quadratic interpolation
  if $|f'(q)| < \epsilon$ **then** break
  find V-combination among (a, p, q, b)
  optional: if progress is small, use Golden Section step
end
return $x^* = q$

Algorithm 3: Quadratic interpolation line search

Quadratic interpolation line search is a good choice for functions behaving like a quadratic function close to their minimum. With a relatively small computational complexity (comparable to bisection), it is possible to achieve fast convergence rate. To make any statements about this, let us first formalize what is meant by *convergence rate*:

Definition. *An iterative procedure producing a sequence $\{x_k\}$ of iterates is said to converge* linearly *if*

$$f(x_k) - f(x^*) \le c_k (f(x_{k-1}) - f(x^*)),$$

with c_k being a constant such that $c_k \to c$. If $c = 0$, the procedure is said to converge super-linearly*.*

Property. *Quadratic interpolation line search converges super-linearly.*

For the sake of the present discussion, we will limit our attention to this statement without proving or justifying it. In the sequel, we will dedicate some attention to convergence rates of different optimization algorithms and way to accelerate them.

4 Cubic interpolation

Another commonly used interpolation technique is cubic interpolation. A cubic function of the form $h(x) = \alpha x^3 + \beta x^2 + \gamma x + \delta$ is used to approximate the function $f(x)$ on the interval $[a, b]$. Since the function has four parameters, we need to specify four equations, which can be obtained by using the values of the function and its derivative at the interval endpoints: $f(a) = h(a)$, $f(b) = h(b)$, $f'(a) = h'(a)$, and $f'(b) = h'(b)$.

Exercise 2. *Derive a closed-form expression for the minimizer of the cubic function $h(x)$ in terms of $f(a)$, $f(b)$, $f'(a)$, and $f'(b)$.*

As before, a V-combination has to be enforced in order to guarantee that the cubic function has a single minimizer inside $[a, b]$. In this case, the points a, b are said to be a V-combination if $f'(a) < 0$ and $f'(b) > 0$. Once the cubic model parameters are found, its minimizer q is guaranteed to be inside the interval $[a, b]$. If $f'(q) < 0$, the minimizer x^* of f is guaranteed to be in $[q, b]$; otherwise, if $f'(q) > 0$, the interval can be restricted to $[a, q]$. This leads to the following iterative procedure:

input : function $f(x)$ and its derivative $f'(x)$; search interval $[a, b]$ forming a V-combination; maximum number of iterations N; tolerance parameter ϵ
output: (approximate) minimizer x^* of f
for $i = 1, 2, \ldots, N$ **do**
  calculate minimizer q of the cubic interpolation
  if $|f'(q)| < \epsilon$ **then** break
  if $f'(q) < 0$ **then** set $a = q$ **else** set $b = q$
  optional: if progress is small, use Bisection step
end
return $x^* = q$

Algorithm 4: Cubic interpolation line search

Note that if the function f is very flat at one of the interval endpoints ($f'(a)$ or $f'(b)$ is close to 0), the minimizer q of the cubic function might be very close to one of the interval endpoints. As the result, the interval $[a, b]$ will not shrink at a sufficient rate. In order to combat this, a bisection step (guaranteeing halving of the interval) is often employed if $1 - \Delta_i/\Delta_{i-1}$ is not large enough.

5 Armijo rule

The techniques discussed so far attempted to solve the one-dimensional minimization problem *exactly*. However, in many practical cases, it is often sufficient to find a point that "sufficiently decreases" the value of the function. Line search methods that find such a point are known as *inexact*.

Let us consider the minimization of $f(x)$ on the ray $x \geq 0$; for convenience, we define a new function $h(x) = f(x) - f(0)$. The function $h(x)$ starts at the origin, goes down with the slope $g = h'(0) = f'(0)$, and then behaves arbitrarily, perhaps even assuming positive values for some x's. We can declare that the function "decreases sufficiently" if its graph falls between the tangent $y = gx$, and another, less steep line, $y = \sigma gx$ for $\sigma \in (0, 1)$. Starting with an initial value $x = x_0$, we can gradually decrease it until the latter condition holds. This gives rise to the iterative procedure known as *backtracking line search* or *Armijo rule*:

input : function $f(x)$ and its derivative $f'(x)$; initial point x; parameters $\sigma \in (0, 1)$ and $\beta \in (0.1)$
output: x such that $f(x)$ has sufficiently decreased compared to $f(0)$
while $f(x) - f(0) > \sigma f'(x)x$ **do** $x \leftarrow \beta x$

Algorithm 5: Backtracking inexact line search (Armijo rule)

Typical values are $\sigma \approx 0.3$, $\beta \approx 0.2$. Bigger values of σ make the search more conservative (at the expense of complexity); smaller values of β make it more aggressive (at the expense of accuracy).