A good reference for the material in this section is Boyd and Vandenberghe's *Convex Optimization*.

Descent methods for unconstrained optimization

Let us consider the general problem of minimizing an *unconstrained* function $f(z) : \mathbb{R}^n \to \mathbb{R}$

$$\min_{z \in \mathbb{R}^n} \; f(z)$$

where $f(z)$ is *convex* and twice differentiable in every direction. This means that $f(\cdot)$ has a *gradient* vector $\nabla f(z)$ and Hessian matrix $\nabla^2 f(z)$ defined at every point z:

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial z_1} \\ \frac{\partial f}{\partial z_2} \\ \vdots \\ \frac{\partial f}{\partial z_N} \end{bmatrix} \qquad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial z_1^2} & \frac{\partial^2 f}{\partial z_1 \partial z_2} & \cdots & \frac{\partial^2 f}{\partial z_1 \partial z_n} \\ \frac{\partial^2 f}{\partial z_1 \partial z_2} & \frac{\partial^2 f}{\partial z_2^2} & \cdots & \frac{\partial^2 f}{\partial z_2 \partial z_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial z_1 \partial z_n} & \frac{\partial^2 f}{\partial z_2 \partial z_n} & \cdots & \frac{\partial^2 f}{\partial z_n^2} \end{bmatrix}$$

Since $f(z)$ is convex, the Hessian $\nabla^2 f(z)$ is always *positive-definite*:

$$v^T [\nabla^2 f(z)] v > 0 \qquad \text{for all} \;\; v \in \mathbb{R}^n.$$

A necessary and sufficient condition for $z^\star$ to be the solution to this program is that

$$\nabla f(z^\star) = 0$$

(i.e. the gradient/derivative is equal to zero at $z^\star$). We will search for such a point iteratively, starting at an initial guess $x^{(0)}$, then moving to $z^{(1)}$, $z^{(2)}$, etc. until ...

In general, we move from $z^{(k)}$ to $z^{(k+1)}$ by first choosing a direction $\Delta^{(k)}$ in which to move, and then setting

$$z^{(k+1)} = z^{(k)} + \alpha^{(k)} \Delta^{(k)}$$

for some appropriate choice of stepsize $\alpha^{(k)}$. Usually, $\alpha^{(k)}$ is chosen adaptively using a line search along the direction $\Delta^{(k)}$ looking for the stepsize which decreases $f(\cdot)$ by a certain amount. Choosing

$$\Delta^{(k)} = -\nabla f(z^{(k)})$$

means that we are moving in the direction of "steepest descent" at each iteration. (This is also called the "gradient descent method").

Newton's method

We have seen how solving a unconstrained quadratic problem of the form

$$\min_{z \in \mathbb{R}^n} \; b^T z + \frac{1}{2} z^T H z$$

corresponds to solving a system of equations — the solution to the above is $z^\star = -H^{-1}b$. Newton's method minimizes a general function

$$\min_{z \in \mathbb{R}^n} \; f(z)$$

by forming a sequence of *quadratic approximations* to f, and solving these using known methods for linear systems. In particular, at a fixed point z_0, we make the *Taylor approximation*

$$f(z_0 + v) \;\approx\; f(z_0) + \nabla f(z_0)^T v + \frac{1}{2} v^T \nabla^2 f(z_0) v. \qquad (1)$$

We choose the direction that minimizes the expression — if the current iterate is $z^{(k)}$, we take

$$b = -\nabla f(z^{(k)})$$
$$H = \nabla^2 f(z^{(k)})$$

and solve for

$$\Delta^{(k)} = H^{-1} b. \qquad (2)$$

Since the Hessian H is positive definite, we can solve this subproblem using conjugate gradients.

Another interpretation of the Newton iteration is that it is *linearizing* the optimality condition. Since f is convex and smooth, we know that a necessary and sufficient condition for $z^\star$ to be a minimizer is $\nabla f(z^\star) = 0$. If we are at iterate $z^{(k)}$, we can use (1) to get

$$\nabla_v f(z^{(k)} + v) \approx \nabla_z f(z^{(k)}) + \nabla^2 f(z^{(k)})v,$$

and of course setting the above equal to zero results in the Newton step (2).

(Newton decrement to guarantee ϵ-optimality...)

Self-concordant functions

There is a special class of functions, called *self-concordant* functions, that are particularly well-suited for Newton's method, both in theory and in practice.

A self-concordant (convex) function $f(t) : \mathbb{R} \to \mathbb{R}$ obeys

$$|f'''(t)| \leq 2f''(t)^{3/2}. \tag{3}$$

The constant 2 above is somewhat arbitrary — the important thing above is that the magnitude of the third derivative is being bounded at every point by the second derivative (recall that if f is convex, the second derivative is always positive). Of course, all linear and quadratic functions are self-concordant.

There is one more important example which we will use below: $f(t) = -\log(t)$. Since $f''(t) = t^{-2}$ and $f'''(t) = -2/t^3$, we see that (3) holds with equality. It should also not be hard to see that $-\log(at + b)$ for all $a, b \in \mathbb{R}$ is also self-concordant. Finally, the sum of two (or more) self-concordant functions is also self-concordant.

For functions on $\mathbb{R}^n$, we simply say f is self-concordant if (3) holds along every line. Self-concordant functions have the very nice property that at any point z, we can get bounds on all of the derivatives (uniform over t) of $h(t) = f(z + tv)$ for any direction v — these bounds can be leveraged to get lower bounds on the $f(z+tv)$ itself. This, in turn, let's you know how close to optimal a certain point is. Precisely, we know that

$$f(z) - f(z^\star) \;\leq\; \lambda(z),$$

where $\lambda(z)$ is called the *Newton decrement*:

$$\lambda(z) = \sqrt{\nabla f(z)^T (\nabla^2 f(z))^{-1} \nabla f(z)}.$$

Notice that $\lambda(z)$ is easy to compute once we have calculated the Newton step direction (2).

This bound on the suboptimality of any given point can be use to derive convergence guarantees for Newton's method on self-concordant functions. There is a number $\gamma < 1/4$ (which itself depends on the particular way in which the line search at each iteration is being performed) such that the number of iterations for $f(z^{(k)})$ to be within ϵ of $f(z^\star)$ can be upper bounded by

$$\frac{f(z^{(0)}) - f(z^\star)}{\gamma} + \log_2 \log_2(1/\epsilon).$$

The $\log_2 \log_2(1/\epsilon)$ can be bounded by a constant (say 6) for all intents and purposes.

Constrained optimization

Let us consider the general problem of minimizing a (smooth, convex) functional $f_0(z)$ with inequality constraints

$$\min_{z \in \mathbb{R}^n} \; f_0(z) \quad \text{subject to} \quad f_i(z) \leq 0, \quad i = 1, \ldots, m. \tag{4}$$

There are some essential ideas from *duality theory* for convex optimization which will help us understand the solution of this program.

We define the *Lagrange function* for the constrained optimization program above as follows:

$$L(z, \lambda) = f_0(z) + \sum_{i=1}^{m} \lambda_i f_i(z), \quad \text{where } \lambda_i \geq 0 \text{ for all } i.$$

It is a fact that $\hat{z}$ is a solution to (4) if there exists $\lambda = \{\lambda_i\}_{i=1}^m$ with $\lambda_{\geq} 0$ such that

$$\nabla_z L(z, \lambda) = \nabla f_0(z) + \sum_{i=1}^{m} \lambda_i \nabla f_i(z) = 0.$$

It is also an easy fact that if $\mathcal{Z}$ is the set of feasible vectors

$$\mathcal{Z} = \{z \in \mathbb{R}^n \ : \ f_i(z) \leq 0 \ i = 1, \ldots, m\},$$

then for any fixed set of Lagrange multipliers $\lambda_i \geq 0$

$$\min_{z \in \mathcal{Z}} L(z, \lambda) \leq \min_{z \in \mathcal{Z}} f_0(z) = f_0(\hat{z}).$$

This simply follows from the fact that for feasible z and positive λ_i,

$$\sum_{i=1}^{m} \lambda_i f_i(z) \leq 0.$$

But now the quantity $\min_{z \in \mathcal{Z}} L(z, \lambda)$ serves as a *lower bound* for the optimal value of our optimization program — of course, different values of λ will give us different lower bounds. The function

$$g(\lambda) = \min_{z \in \mathcal{Z}} L(z, \lambda)$$

is called the *Lagrange dual function* for the problem (4).

Log-barrier methods

There is a general method for solving (4) by transforming it into a series of *unconstrained* problems. First, suppose that $I_-(u)$ is a perfect "barrier function"

$$I_-(u) = \begin{cases} 0 & u \leq 0 \\ \infty & u > 0 \end{cases}.$$

Then we could re-write the constrained program as

$$\min_{z \in \mathbb{R}^n} \; f_0(z) + \sum_{i=1}^{m} I_-(f_i(z)). \tag{5}$$

The solutions to (5) ad (4) do indeed coincide, but the problem is that the functional in (5) is not smooth enough (i.e. not twice differentiable) to apply Newtons method. This is addressed by approximating the barrier function with a smooth function.

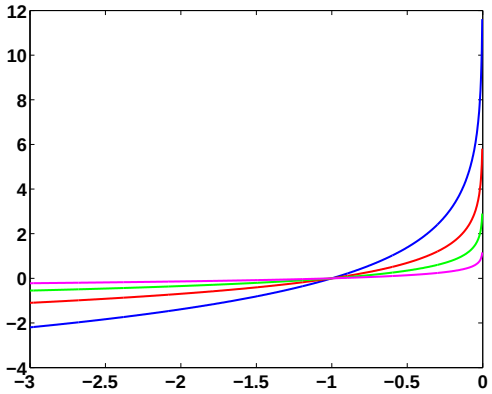
Instead of the sharp barrier function above, we might use a *log barrier*:

$$\hat{I}_-(u) = -\frac{1}{\tau}\log(-u).$$

Notice that $\hat{I}_-(u) \to \infty$ as $u \to 0$ from the left ($u < 0$). We can think of $\hat{I}_-(u)$ as an approximation to the sharp barrier $I_-(u)$, this approximation gets better for larger τ:

$$-\frac{1}{\tau}\log(-u) \to I_-(u) \quad \text{as} \;\; \tau \to \infty.$$

Below, we see $-\frac{1}{\tau}\log(-u)$ plotted for $\tau = 1/2, 1, 2$, and 5 (the blue, red, green, and magenta lines, respectively):



So we can approximate the unconstrained program in (4) by the unconstrained program

$$\min_{z \in \mathbb{R}^n} \; f_0(z) - \frac{1}{\tau} \sum_{i=1}^{m} \log(-f_i(z)). \tag{6}$$

The basic idea is to solve this for a particular value of τ using Newton's method, then increase τ, and then solve again. To apply Newton's method, we will need the gradient and the Hessian of

$$b(z) = -\sum_{i=1}^{m} \log(-f_i(z)).$$

They can be computed using the chain rule; the results are:

$$\nabla b(z) = -\sum_{i=1}^{m} \frac{1}{f_i(z)} \nabla f_i(z) \tag{7}$$

$$\nabla^2 b(z) = \sum_{i=1}^{m} \frac{1}{f_i(z)^2} \nabla f_i(z) \nabla f_i(z)^T - \sum_{i=1}^{m} \frac{1}{f_i(z)} \nabla^2 f_i(z). \tag{8}$$

Also, it is a fact that if $f(z)$ is self-concordant, then $f(z) + (1/\tau)b(z)$ is also self-concordant, and we have an analytic bound on the number of Newton steps required to converge to a certain precision.

After solving (6) for a particular value of τ, we can actually guarantee that the current solution is close to being optimal to the

constrained problem (4). Let z^τ be the solution to (6). We know that it must be the case that

$$\begin{aligned} 0 &= \nabla f_0(z^\tau) + \frac{1}{\tau}\nabla b(z^\tau) \\ &= \nabla f_0(z^\tau) - \frac{1}{\tau}\sum_{i=1}^{m}\frac{1}{f_i(z^\tau)}\nabla f_i(z^\tau) \\ &= \nabla f_0(z^\tau) + \sum_{i=1}^{m}\lambda_i^\tau \nabla f_i(z^\tau), \end{aligned}$$

where

$$\lambda_i^\tau = \frac{-1}{\tau f_i(z^\tau)} > 0.$$

The above also tells us that the Lagrange function:

$$L(z,\lambda) = f_0(z) + \sum_{i=1}^{m}\lambda_i f_i(z)$$

is also minimized at z^τ:

$$\min_{z\in\mathbb{Z}} L(z,\lambda^\tau) = f_0(z^\tau) + \sum_{i=1}^{m}\lambda_i^\tau f_i(z^\tau).$$

Thus, we know that

$$f_0(z^\tau) + \sum_{i=1}^{m}\lambda_i^\tau f_i(z^\tau) \;\leq\; f_0(\hat{z}),$$

which means

$$\begin{aligned} f_0(z^\tau) - f_0(\hat{z}) &\leq -\sum_{i=1}^{m}\lambda_i^\tau f_i z(\tau) \\ &= \sum_{i=1}^{m}\frac{1}{\tau} \\ &= \frac{m}{\tau}. \end{aligned}$$

So the solution to (6) is within m/τ of optimal.

One inference we can make from this discussion is that if we solve (6) once with $\tau = m/\epsilon$:

$$\min_z \; f_0(z) - \frac{\epsilon}{m} \sum_{i=1}^{m} \log(-f_i(z)),$$

then the solution of the above will be within ϵ of the solution to (4). The problem, though, is that if ϵ is small (which we want it to be), the sides of the barrier function are very steep and the middle is flat, which causes a number of practical problems.

Instead, we start with a small value of τ (usually $\tau = 1$), solve (6), increase τ, then repeat. The idea is that for large values of τ, the solution will not move much a τ increases, which means each step will take only a small number of Newton iterations.

Here is the *log barrier method:*
Start with a feasible z_{start}, $\tau = \tau_0$, a tolerance $\epsilon > 0$, and a parameter μ

1. Solve (6) using z_{start} as the starting point. Call the solution $\hat{z}$.
2. If $m/\tau < \epsilon$, then stop, returning $\hat{z}$.
3. Otherwise, set $z_{\text{start}} = \hat{z}$ and $\tau = \mu\tau$ and goto 1.

Thanks to our bound on how closely the solution to (6) matches that of (4), we can predict the number of steps the log barrier method will need in advance. After the kth step, the current log barrier solution should be within $m/(\mu^k \tau_0)$ of the optimal value of (4). Thus for a desired accuracy of ϵ, we will need

$$\text{total barrier steps} \;=\; \left\lceil \frac{\log(m/(\epsilon\tau_0))}{\log \mu} \right\rceil .$$

Some comments on choosing these parameters:

- The overall performance of the program is robust to the choice of μ. Smaller values of μ mean that there will be more barrier iterations, but Newton's algorithm (to solve the barrier problem) will converge more quickly, basically because its starting point will be better. Larger values of μ mean that there will be fewer barrier iterations, but Newton's method will take longer to solve each subproblem. In practice, $\mu = 10$ or 20 seems to work for many problems.
- The value of τ_0 can make a difference. We would like to use as large a value as possible, but we do not want to make the initial problem too hard. In practice, choosing a τ_0 on the order of $m/f(z_{\text{start}})$ seems to work.

Example: ℓ_1 minimization with quadratic constraints

Consider the problem

$$\min_{x \in \mathbb{R}^N} \|x\|_1 \quad \text{subject to} \quad \|\Phi x - y\|_2 \leq \epsilon.$$

Solving this problem directly is problematic, since the functional $\|x\|_1$ is not smooth. But by adding some dummy variables, we can convert this into a smooth program:

$$\begin{aligned} \min_{u, x \in \mathbb{R}^N} \sum_{n=1}^{N} u[n] \quad \text{subject to} \quad & x - u \leq 0, \\ & -x - u \leq 0, \\ & \frac{1}{2}\left(\|\Phi x - y\|_2^2 - \epsilon^2\right) \leq 0. \end{aligned}$$

We can align the notation with the discussion in previous sections of these notes by taking

$$z = \begin{bmatrix} x \\ u \end{bmatrix}.$$

We now have a (smooth, convex) optimization program with $n = 2N$ variables and $m = 2N + 1$ inequality constraints — $2N$ of these are linear constraints, and the last one is *conic*. Define the functions

$$f_{u_1} = x - u, \quad f_{u_2} = -x - u, \quad f_\epsilon = \frac{1}{2}\left(\|\Phi x - y\|_2^2 - \epsilon^2\right),$$

and notice that f_{u_1} and f_{u_2} are N-vectors with each entry only dependent on one value in x and one value in u, while f_ϵ is a scalar that depends on every entry in x. At a fixed point (x, u),

let $r = \Phi x - y$. Then for a given τ, we can write the Newton step (as in (2)) for (6) (using (7) and (8)) as

$$\begin{bmatrix} \Sigma_{11} - f_\epsilon^{-1} A^T A + f_\epsilon^{-2} A^T r r^T A & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{11} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} = \begin{bmatrix} f_{u_1}^{-1} - f_{u_2}^{-1} + f_\epsilon^{-1} A^T r \\ -\tau \mathbf{1} - f_{u_1}^{-1} - f_{u_2}^{-1} \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

where

$$\begin{aligned} \Sigma_{11} &= F_{u_1}^{-2} + F_{u_2}^{-2} \\ \Sigma_{12} &= -F_{u_1}^{-2} + F_{u_2}^{-2}, \end{aligned}$$

and $F_{u_1}^{-1}$ is a diagonal matrix with f_u^{-1} along the diagonal (and similarly for $F_{u_2}^{-1}$).

Complementary slackness and the KKT conditions

We have used three different programs to discuss a general convex optimization program;
the *primal*

$$\min_z \; f(z) \quad \text{subject to} \quad f_i(z) \leq 0, \; i = 1, \ldots, m,$$

the *Lagrange function*

$$L(z, \lambda) = f(z) + \sum_{i=1}^{m} \lambda_i f_i(z), \quad \lambda_i \geq 0,$$

and the *Lagrange dual*

$$g(\lambda) = \min_{z \in \mathcal{Z}} L(z, \lambda) = \min_{z \in \mathcal{Z}} \left(f(z) + \sum_{i=1}^{m} \lambda_i f_i(z) \right).$$

We have seen that

$$g(\lambda) \leq f(z) \quad \text{for all } \lambda_i \geq 0, \text{ and feasible } z, \; f_i(z) \leq 0.$$

Thus we always have

$$\max_{\lambda \geq 0} g(\lambda) \;\leq\; \min_z \; f(z) \quad \text{subject to} \quad f_i(z) \leq 0, \; i = 1, \ldots, m$$

Under mild conditions, the solutions of these programs are equal. If the primal is minimized at $z^\star$ and the dual is maximized at $\lambda^\star$, we have

$$f(z^\star) = g(\lambda^\star).$$

But since

$$g(\lambda^\star) = \min_{z \in \mathcal{Z}} \left(f(z) + \sum_{i=1}^{m} \lambda_i^\star f_i(z) \right) \leq f(z^\star) + \sum_{i=1}^{m} \lambda_i^\star f_i(z^\star) \leq f(z^\star),$$

it must be the case that

$$f(z^\star) + \sum_{i=1}^{m} \lambda_i^\star f_i(z^\star) = f(z^\star),$$

meaning that

$$\sum_{i=1}^{m} \lambda_i^\star f_i(z^\star) = 0.$$

Since $\lambda_i^\star \geq 0$ and $f_i(z^\star) \leq 0$, this means that

$$\begin{aligned} \lambda_i^\star > 0 &\Rightarrow f_i(z^\star) = 0 \\ f_i(z^\star) < 0 &\Rightarrow \lambda_i^\star = 0, \end{aligned}$$

or more succinctly

$$\lambda_i^\star f_i(z^\star) = 0 \quad \text{for all } i = 1, 2, \ldots, m.$$

This gives us the final piece of the puzzle we need to state the Karesh-Kuhn-Tucker conditions for optimality. Any solution $z^\star$ to the primal and solution $\lambda^\star$ to the dual will obey

$$\begin{aligned} f_i(z^\star) \leq 0, &\quad i = 1, \ldots, m \quad \text{(primal feasibility)} \\ \lambda_i^\star \geq 0, &\quad i = 1, \ldots, m \quad \text{(dual feasibility)} \\ \lambda_i^\star f_i(z^\star) = 0, &\quad i = 1, \ldots, m \quad \text{(complementary slackness)} \end{aligned}$$

$$\nabla f(z^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(z^\star) = 0.$$

These conditions are also sufficient.

The moral of the story is you can reduce any convex optimization program to solving a system of (nonlinear) equations in z and λ. In fact, there is a whole suite of techniques call *primal-dual* algorithms which try to do this directly (again, using Newton's method iteratively to linearize the equations, then solve).