WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Optimization and Root Finding

Rainer Hirk

2021-11-22

Root finding and unconstrained smooth optimization are closely related:

- Solving $f(x) = 0$ can be accomplished via minimizing $\|f(x)\|^2$

Basics

Root finding and unconstrained smooth optimization are closely related:

- Solving $f(x) = 0$ can be accomplished via minimizing $\|f(x)\|^2$
- Unconstrained optima of f must be critical points, i.e., solve $\nabla f(x) = 0$

(Note: f scalar for optimization and typically vector-valued for root finding.)

Linear equations and linear least squares problems can be solved "exactly" using techniques from numerical linear algebra.

Otherwise: solve "approximately" as limits of iterations $x_{k+1} = g(x_k)$.

Fixed Point Iteration

Consider iteration $x_{k+1} = g(x_k)$ with limit x^* and g smooth. Then $x^* = g(x^*)$ and

$$x_{k+1} = g(x_k) \approx g(x^*) + J_g(x^*)(x_k - x^*) = x^* + J_g(x^*)(x_k - x^*)$$

where

$$J_g(x) = \left[\frac{\partial g_i}{\partial x_j}(x)\right]$$

is the Jacobian of g at x (sometimes also written as $(\nabla g)'(x)$). Thus:

$$x_{k+1} - x^* \approx J_g(x^*)(x_k - x^*)$$

Fixed Point Iteration

In general, for local convergence it is necessary and sufficient that

$$\rho(J_g(x^*)) < 1$$

where

$$\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$$

is the *spectral radius* of A.

In this case, we get (at least) linear (local) convergence, i.e.,

$$\|x_{k+1} - x^*\| \leq C\|x_k - x^*\|$$

for some $0 < C < 1$ and k sufficiently large.

Outline

Newton's Method

Suppose we have an approximation x_k to x^* and that in a neighborhood of x_k, the linearization

$$L_k(x) = f(x_k) + J_f(x_k)(x - x_k)$$

is a good approximation to f.

An obvious candidate for a next approximation is obtained by solving $L_k(x) = 0$, i.e.,

$$x_{k+1} = x_k - J_f(x_k)^{-1} f(x_k) = g(x_k), \qquad g(x) = x - J_f(x)^{-1} f(x),$$

This is the "mathematical" form: the computational one is

$$\text{Solve } J_f(x_k)s_k = -f(x_k) \text{ for } s_k; \qquad x_{k+1} = x_k + s_k.$$

Iteration function g has components and partials

$$\begin{aligned} g_i(x) &= x_i - \sum_l [J_f(x)^{-1}]_{il} f_l(x), \\ \frac{\partial g_i}{\partial x_j}(x) &= \delta_{ij} - \sum_l \frac{\partial [J_f(x)^{-1}]_{il}}{\partial x_j} f_l(x) - \sum_l [J_f(x)^{-1}]_{il} \frac{\partial f_l}{\partial x_j}(x). \end{aligned}$$

As $f(x^*) = 0$,

$$\frac{\partial g_i}{\partial x_j}(x^*) = \delta_{ij} - \sum_l [J_f(x^*)^{-1}]_{il} \frac{\partial f_l}{\partial x_j}(x^*) = \delta_{ij} - [J_f(x^*)^{-1} J_f(x^*)]_{ij} = 0$$

so that $J_g(x^*)$ vanishes! Thus, local convergence is *super-linear* (and can in fact be shown to be at least quadratic) in the sense that

$$\|x_{k+1} - x^*\| \leq \alpha_k \|x_k - x^*\|, \qquad \lim_k \alpha_k = 0.$$

Newton's Method: Discussion

Due to super-linear convergence, works very nicely once we get close enough to a root where the above approximations apply.

However (with n the dimension of the problem):

- In each step, $O(n^2)$ derivatives needs to be computed (exactly or numerically), which can be costly (in particular if function and/or derivative evaluations are costly);

Newton's Method: Discussion

Due to super-linear convergence, works very nicely once we get close enough to a root where the above approximations apply.

However (with n the dimension of the problem):

- In each step, $O(n^2)$ derivatives needs to be computed (exactly or numerically), which can be costly (in particular if function and/or derivative evaluations are costly);
- In each step, an $n \times n$ linear system needs to be solved, which takes $O(n^3)$ operations.

Newton's Method: Discussion

Due to super-linear convergence, works very nicely once we get close enough to a root where the above approximations apply.

However (with n the dimension of the problem):

- In each step, $O(n^2)$ derivatives needs to be computed (exactly or numerically), which can be costly (in particular if function and/or derivative evaluations are costly);
- In each step, an $n \times n$ linear system needs to be solved, which takes $O(n^3)$ operations.
- For super-linear convergence the exact Jacobian is not needed.

Super-Linear Convergence

Attractive because "faster than linear". Also,

$$|\|x_{k+1} - x_k\| - \|x_k - x^*\|| \leq \|x_{k+1} - x^*\|$$

so that with super-linear convergence,

$$\left| \frac{\|x_{k+1} - x_k\|}{\|x_k - x^*\|} - 1 \right| \leq \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = \alpha_k \to 0.$$

I.e., relative change in the x_k update is a good approximation for the relative error in the approximation of x^* by x_k (which is commonly used for stopping criteria).

Super-Linear Convergence

One can show: for sequences

$$x_{k+1} = x_k - B_k^{-1} f(x_k)$$

with suitable non-singular matrices B_k one has super-linear (local) convergence to x^* with $f(x^*) = 0$ if and only if

$$\lim_{k \to \infty} \frac{\|(B_k - J_f(x^*))(x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} = 0,$$

i.e., if B_k converges to $J_f(x^*)$ along the directions $s_k = x_{k+1} - x_k$ of the iterative method.

Suggests investigating iterative "Quasi-Newton" methods with such B_k which are less costly to compute and/or invert.

Broyden's Method

Note that Newton's method is based on the approximation

$$f(x_{k+1}) \approx f(x_k) + J_f(x_k)(x_{k+1} - x_k).$$

Suggests considering approximations B_{k+1} which exactly satisfy the *secant equation*

$$f(x_{k+1}) = f(x_k) + B_{k+1}(x_{k+1} - x_k).$$

Of all such B_{k+1}, the one with the least change to B_k seems particularly attractive. I.e., solutions to

$$\|B - B_k\|_F \to \min, \qquad y_k = f(x_{k+1}) - f(x_k) = Bs_k$$

(where $\|\cdot\|_F$ is the Frobenius norm).

Broyden's Method

We have

$$\|B - B_k\|_F^2 = \|\mathrm{vec}(B) - \mathrm{vec}(B_k)\|^2, \qquad \mathrm{vec}(Bs_k - y_k) = (s_k' \otimes I)\mathrm{vec}(B) - y_k.$$

Thus, writing

$$b = \mathrm{vec}(B), \quad b_k = \mathrm{vec}(B_k), \quad A_k = s_k' \otimes I,$$

optimization problem is

$$\|b - b_k\|^2 \to \min, \quad A_k b = y_k.$$

Convex quadratic optimzation problem with linear constraints: can solve using geometric ideas (orthogonal projections on affine subspaces) or using Lagrange's method.

Broyden's Method

Lagrangian:

$$L(b, \lambda) = \frac{1}{2}\|b - b_k\|^2 + \lambda'(A_k b - y_k).$$

For critical point:

$$\nabla_b L = b - b_k + A_k'\lambda = 0, \qquad \nabla_\lambda L = A_k b - y_k = 0.$$

Thus $b = b_k - A_k'\lambda$ with $y_k = A_k b = A_k(b_k - A_k'\lambda)$, so $\lambda = (A_k A_k')^{-1}(A_k b_k - y_k)$ and

$$b = b_k - A_k'(A_k A_k')^{-1}(A_k b_k - y_k).$$

Broyden's Method

As $A_k A_k' = (s_k' \otimes I)(s_k \otimes I) = s_k' s_k I$ and $A_k b_k = (s_k' \otimes I)\mathrm{vec}(B_k) = \mathrm{vec}(B_k s_k)$,

$$A_k'(A_k A_k')^{-1}(A_k b_k - y_k) = \frac{1}{s_k' s_k}(s_k \otimes I)(A_k b_k - y_k) = \frac{1}{s_k' s_k}\mathrm{vec}((A_k b_k - y_k)s_k')$$

and hence

$$\mathrm{vec}(B) = b = \mathrm{vec}(B_k) - \mathrm{vec}\left(\frac{(A_k b_k - y_k)s_k'}{s_k' s_k}\right) = \mathrm{vec}\left(B_k - \frac{(B_k s_k - y_k)s_k'}{s_k' s_k}\right)$$

and thus

$$B = B_k + \frac{(y_k - B_k s_k)s_k'}{s_k' s_k}$$

Broyden's Method

Based on a suitable guess x_0 and a suitable full rank approximation B_0 for the Jacobian (e.g., I), iterate using

$$\begin{aligned} &\textit{Solve } B_k s_k = -f(x_k) \textit{ for } s_k \\ x_{k+1} &= x_k + s_k \\ y_k &= f(x_{k+1}) - f(x_k) \\ B_{k+1} &= B_k + (y_k - B_k s_k) s_k' / (s_k' s_k). \end{aligned}$$

Again, this is the mathematical form.

Computationally, can improve!

Note that iteration for B_k performs *rank-one updates* of the form $B_{k+1} = B_k + u_k v_k'$, and all we need is the *inverse* $H_k = B_k^{-1}$!

Broyden's Method

Remember Sherman-Morrison formula for inverse of rank-one update:

$$(A + uv')^{-1} = A^{-1} - (1/\sigma)A^{-1}uv'A^{-1}, \qquad \sigma = 1 + v'A^{-1}u \neq 0.$$

Thus,

$$H_{k+1} = B_{k+1}^{-1} = (B_k + u_k v_k')^{-1} = H_k - \frac{1}{1 + v_k' H_k u_k} H_k u_k v_k' H_k$$

with $u_k = (y_k - B_k s_k)$ and $v_k = s_k/(s_k' s_k)$ so that

$$H_k u_k = B_k^{-1} u_k = B_k^{-1} y_k - s_k = H_k y_k - s_k$$

and

$$1 + v_k' H_k u_k = 1 + \frac{s_k'(H_k y_k - s_k)}{s_k' s_k} = \frac{s_k' H_k y_k}{s_k' s_k}.$$

Broyden's Method

Computational form: based on suitable guess x_0 and a suitable full rank approximation H_0 for the inverse of the Jacobian (e.g., I), iterate using

$$\begin{aligned} s_k &= -H_k f(x_k) \\ x_{k+1} &= x_k + s_k \\ y_k &= f(x_{k+1}) - f(x_k) \\ H_{k+1} &= H_k - \frac{(H_k y_k - s_k) s_k' H_k}{s_k' H_k y_k} \end{aligned}$$

(of course needs that $s_k' H_k y_k \neq 0$).

Spectral Methods

Used for very large n. E.g., Barzilai-Borwein methods.

One variant (DF-SANE) based on idea to use very simple B_k or H_k proportional to the identity matrix.

However: when $B_k = \beta_k I$, secant condition $B_{k+1}s_k = y_k$ typically cannot be achieved exactly. Instead, determine β_{k+1} to solve

$$\|y_k - B_{k+1}s_k\|^2 = \|y_k - \beta_{k+1}s_k\|^2 \to \min$$

with solution $\beta_{k+1} = y_k' s_k / s_k' s_k$. Gives iteration

$$x_{k+1} = x_k - f(x_k)/\beta_k, \qquad \beta_{k+1} = y_k' s_k / s_k' s_k$$

or equivalently (with s_k and y_k as before),

$$x_{k+1} = x_k - \alpha_k f(x_k), \qquad \alpha_{k+1} = s_k' s_k / y_k' s_k.$$

Alternatively, when using $H_k = \alpha_k I$, the exact secant condition would be $H_{k+1} y_k = s_k$, and one determines α_{k+1} to solve

$$\|s_k - H_{k+1} y_k\|^2 = \|s_k - \alpha_{k+1} y_k\|^2 \to \min$$

with solution $\alpha_{k+1} = s_k' y_k / y_k' y_k$. Gives iteration

$$x_{k+1} = x_k - \alpha_k f(x_k), \qquad \alpha_{k+1} = \frac{s_k' y_k}{y_k' y_k}.$$

Root Finding with R

Methods only only available in add-on packages.

- Package nleqslv has function `nleqslv()` providing the Newton and Broyden methods with a variety of global strategies.

Root Finding with R

Methods only only available in add-on packages.

- Package nleqslv has function `nleqslv()` providing the Newton and Broyden methods with a variety of global strategies.
- Package BB has function `BBsolve()` providing Barzilai-Borwein solvers, and and `multiStart()` for starting solvers from multiple starting values.

Root Finding with R

Methods only only available in add-on packages.

- Package nleqslv has function `nleqslv()` providing the Newton and Broyden methods with a variety of global strategies.
- Package BB has function `BBsolve()` providing Barzilai-Borwein solvers, and and `multiStart()` for starting solvers from multiple starting values.
- ???

Examples

System

$$x_1^2 + x_2^2 = 2, \qquad e^{x_1 - 1} + x_2^3 = 2$$

Has one solution at $x_1 = x_2 = 1$.

Outline

- Root Finding
- Unconstrained Optimization
- Constrained Optimization
- Optimization with R

Consider unconstrained minimization problem with objective function f. If f is smooth:

$$f(x^* + s) = f(x^*) + \nabla f(x^*)'s + \frac{1}{2}s'H_f(x^* + \alpha s)s$$

for some $0 < \alpha < 1$, with $H_f(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right]$ the Hessian of f at x. Hence:

- x^* local minimum $\Longrightarrow \nabla f(x^*) = 0$ (x^* is a critical point of f) and $H_f(x^*)$ non-negative definite (necessary first and second order conditions)

Consider unconstrained minimization problem with objective function f. If f is smooth:

$$f(x^* + s) = f(x^*) + \nabla f(x^*)'s + \frac{1}{2}s'H_f(x^* + \alpha s)s$$

for some $0 < \alpha < 1$, with $H_f(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right]$ the Hessian of f at x. Hence:

- x^* local minimum $\Longrightarrow \nabla f(x^*) = 0$ (x^* is a critical point of f) and $H_f(x^*)$ non-negative definite (necessary first and second order conditions)
- x^* a critical point of f and $H_f(x^*)$ positive definite (sufficient second order condition) $\Longrightarrow x^*$ local minimum

Steepest Descent

Linear approximation

$$f(x + \alpha s) \approx f(x) + \alpha \nabla f(x)' s$$

suggests looking for good α along the steepest descent direction $s = -\nabla f(x)$.

Typically performed as

$s_k = -\nabla f(x_k)$
Choose α_k *to minimize* $f(x_k + \alpha s_k)$
$x_{k+1} = x_k + \alpha_k s_k$.

(method of steepest descent [with line search]).

Newton's Method

Use local quadratic approximation

$$f(x+s) \approx f(x) + \nabla f(x)'s + \frac{1}{2}s'H_f(x)s$$

(equivalently, linear approximation for ∇f) and minimize RHS over s to obtain Newton's method

$$x_{k+1} = x_k - H_f(x_k)^{-1}\nabla f(x_k)$$

with computational form

$$\text{Solve } H_f(x_k)s_k = -\nabla f(x_k) \text{ for } s_k; \qquad x_{k+1} = x_k + s_k.$$

Quasi-Newton Methods

Similarly to the root-finding case, there are many modifications of the basic scheme, the general form

$$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k)$$

referred to as *quasi-Newton* methods: adding the α_k allows to improve the global performance by controlling the step size.

(Note however that away from a critical point, the Newton direction s does not necessarily result in a local decrease of the objective function.)

Quasi-Newton Methods

Similar to the root-finding case, one can look for simple updating rules for Hessian approximations B_k subject to the secant constraint

$$B_k s_k = y_k = \nabla f(x_{k+1}) - \nabla f(x_k),$$

additionally taking into account that the B_k should be symmetric (as approximations to the Hessians). This motivates solving

$$\|B - B_k\|_F \to \min, \qquad B \text{ symmetric}, \quad y_k = B s_k.$$

which can easily be shown to have the unique solution

$$B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)'}{(y_k - B_k s_k)' s_k}.$$

Symmetric rank-one (SR1) update formula (Davidon, 1959).

SR1 has numerical difficulties: one thus considers similar problems with different norms, such as weighted Frobenius norms

$$\|B - B_k\|_{F,W} = \|W^{1/2}(B - B_k)W^{1/2}\|_F.$$

One can show: for any (symmetric) W for which $Wy_k = s_k$ (such as the inverse of the average Hessian $\int_0^1 H_f(x_k + \tau s_k)\, d\tau$ featuring in Taylor's theorem) which makes the approximation problem non-dimensional,

$$\|B - B_k\|_{F,W} \to \min, \qquad B \text{ symmetric}, \quad y_k = Bs_k$$

has the the unique solution

$$(I - \rho_k y_k s_k')B_k(I - \rho_k s_k y_k') + \rho_k y_k y_k', \qquad \rho_k = 1/y_k' s_k.$$

This is the DFP (Davidon-Fletcher-Powell) update formula.

Average Hessian: we have

$$\begin{aligned}\nabla f(x+s) - \nabla f(x) &= \nabla f(x+\tau s)\Big|_0^1 \\ &= \int_0^1 \frac{d}{d\tau} \nabla f(x+\tau s)\, d\tau \\ &= \left(\int_0^1 H_f(x+\tau s)\, d\tau\right) s\end{aligned}$$

and hence (ignoring α_k for simplicity)

$$y_k = \nabla f(x_k + s_k) - \nabla f(x_k) = \left(\int_0^1 H_f(x_k + \tau s_k)\, d\tau\right) s_k.$$

Quasi-Newton Methods

Most effective quasi-Newton formula is obtained by applying the corresponding approximation argument to $H_k = B_k^{-1}$, i.e., by using the unique solution to

$$\|H - H_k\|_{F,W} \to \min \qquad H \text{ symmetric}, \quad s_k = Hy_k.$$

with W any matrix satisfying $Ws_k = y_k$ (such as the average Hessian matrix), which is given by

$$(I - \rho_k s_k y_k')H_k(I - \rho_k y_k s_k') + \rho_k s_k s_k', \qquad \rho_k = \frac{1}{y_k' s_k}.$$

This is the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update formula.

Using Sherman-Morrison-Woodbury formula
$(A+UV')^{-1} = A^{-1} - A^{-1}U(I+V'A^{-1}U)^{-1}V'A^{-1}$ and lengthy computations, one can show:

- BFGS update for H_k corresponds to

$$B_{k+1} = B_k + \frac{y_k y_k'}{y_k' s_k} - \frac{B_k s_k s_k' B_k}{s_k' B_k s_k}$$

Using Sherman-Morrison-Woodbury formula
$(A+UV')^{-1} = A^{-1} - A^{-1}U(I+V'A^{-1}U)^{-1}V'A^{-1}$ and lengthy computations, one can show:

- BFGS update for H_k corresponds to

$$B_{k+1} = B_k + \frac{y_k y_k'}{y_k' s_k} - \frac{B_k s_k s_k' B_k}{s_k' B_k s_k}$$

- DFP update for B_k corresponds to

$$H_{k+1} = H_k + \frac{s_k s_k'}{s_k' y_k} - \frac{H_k y_k y_k' H_k}{y_k' H_k y_k}$$

Rank-two updates. (Note the symmetry.)

Conjugate Gradient Methods

Alternative to quasi-Newton methods. Have the form

$$x_{k+1} = x_k + \alpha_k s_k, \qquad s_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1} s_k$$

where the β are chosen such that s_{k+1} and s_k are suitably conjugate.

E.g., the Fletcher-Reeves variant uses

$$\beta_{k+1} = \frac{g'_{k+1} g_{k+1}}{g'_k g_k}, \qquad g_{k+1} = \nabla f(x_{k+1}).$$

Advantage: require no matrix storage (uses only gradients) and are faster than steepest descent, hence suitable for large-scale optimization problems.

Gauss-Newton Method

Minimizing objective functions of the form

$$\phi(x) = \frac{1}{2}\sum_i r_i(x)^2 = \|r(x)\|^2/2,$$

is known (roughly) as non-linear least squares problems, corresponding to minimizing the sum of squares of residuals $r_i(x) = t_i - f(e_i, x)$ (in statistics, we usually write x_i and y_i for the inputs and targets, and θ instead of x for the unknown parameter). Clearly,

$$\nabla\phi(x) = J_r(x)'r(x), \qquad H_\phi(x) = J_r(x)'J_r(x) + \sum_i r_i(x)H_{r_i}(x).$$

Gauss-Newton Method

Straightforward Newton method typically replaced by the *Gauss-Newton method*

$$(J_r(x_k)' J_r(x_k))s_k = -J_r(x_k)' r(x_k)$$

which drops the terms involving the H_{r_i} from the Hessian based on the idea that in the minimum the residuals r_i should be "small".

Replaces a non-linear least squares problem by a sequence of linear least-squares problems with the right limit.

However, simplification may lead to ill-conditioned or rank deficient linear problems: if so, the *Levenberg-Marquardt method*

$$(J_r(x_k)' J_r(x_k) + \mu_k I)s_k = -J_r(x_k)' r(x_k)$$

with suitably chosen $\mu_k > 0$ can be employed.

Outline

- Root Finding
- Unconstrained Optimization
- Constrained Optimization
- Optimization with R

Local optima are relative to neighborhoods in feasible set X.

A suitably "interior" point x^* in X is a local minimum iff $t \mapsto f(x(t))$ has a local minimum at $t = 0$ for all smooth curves $t \mapsto x(t)$ in X defined in some open interval containing 0 with $x(0) = x^*$.

(Alternatively, if $J_g(x^*)$ has full column rank, use the implicit mapping theorem to obtain a local parametrization $x = x(s)$ of the feasible set with, say, $x(0) = x^*$, and consider the conditions for $f(x(s))$ to have an unconstrained local minimum at $s = 0$.)

We have

$$\begin{aligned}\frac{df(x(t))}{dt} &= \sum_j \frac{\partial f}{\partial x_j}(x(t))\dot{x}_j(t) \\ \frac{d^2f(x(t))}{dt^2} &= \sum_{j,k} \frac{\partial^2 f}{\partial x_j \partial x_k}(x(t))\dot{x}_j(t)\dot{x}_k(t) + \sum_j \frac{\partial f}{\partial x_j}(x(t))\ddot{x}_j(t)\end{aligned}$$

From the first: for local minimum necessarily

$$\langle \nabla f(x^*), s \rangle = 0$$

for all feasible directions s at x^*.

Suppose that X is defined by equality constraints $g(x) = [g_1(x), \ldots, g_m(x)]' = 0$. Then $g_i(x(t)) \equiv 0$ and hence

$$\sum_j \frac{\partial g_i}{\partial x_j}(x(t))\dot{x}_j(t) = 0.$$

Hence: set of feasible directions at x^* is given by $\mathrm{Null}(J_g(x^*))$.

Necessary first order condition translates into:

$$\nabla f(x^*) \in (\mathrm{Null}(J_g(x^*)))^\perp = \mathrm{Range}(J_g(x^*)'),$$

i.e., there exist Lagrange multipliers λ such that

$$\nabla f(x^*) = -J_g(x^*)'\lambda.$$

Differentiating once more,

$$\sum_{j,k} \frac{\partial^2 g_i}{\partial x_j \partial x_k}(x(t))\dot{x}_j(t)\dot{x}_k(t) + \sum_j \frac{\partial g_i}{\partial x_j}(x(t))\ddot{x}_j(t) = 0.$$

Multiply by λ_i from above and add to expression for $d^2 f(x(t))/dt^2$ at $t = 0$:

$$\begin{aligned}
\left.\frac{d^2 f(x(t))}{dt^2}\right|_{t=0} &= \sum_{j,k}\left(\frac{\partial^2 f}{\partial x_j \partial x_k}(x^*) + \sum_i \lambda_i \frac{\partial^2 g_i}{\partial x_j \partial x_k}(x^*)\right) s_j s_k \\
&\quad + \sum_j \left(\frac{\partial f}{\partial x_j}(x^*) + \sum_i \lambda_i \frac{\partial g_i}{\partial x_j}(x^*)\right)\ddot{x}_j(0) \\
&= s'\left(H_f(x^*) + \textstyle\sum_i \lambda_i H_{g_i}(x^*)\right) s \\
&= s' B(x^*, \lambda) s
\end{aligned}$$

For local minimum, must thus also have $s'B(x^*, \lambda)s \geq 0$ for all feasible directions s at x^*, or equivalently, for all $s \in \mathrm{Null}(J_g(x^*)$.

Let Z be a basis for $\mathrm{Null}(J_g(x^*)$. Then:

- x^* constrained local minimum $\implies \nabla f(x^*) = -J_g(x^*)'\lambda$ and projected (reduced) Hessian $Z'B(x^*, \lambda)Z$ non-negative definite

For local minimum, must thus also have $s'B(x^*, \lambda)s \geq 0$ for all feasible directions s at x^*, or equivalently, for all $s \in \mathrm{Null}(J_g(x^*)$.

Let Z be a basis for $\mathrm{Null}(J_g(x^*)$. Then:

- x^* constrained local minimum $\implies \nabla f(x^*) = -J_g(x^*)'\lambda$ and projected (reduced) Hessian $Z'B(x^*, \lambda)Z$ non-negative definite
- $\nabla f(x^*) = -J_g(x^*)'\lambda$ and $Z'B(x^*, \lambda)Z$ positive definite $\implies x^*$ constrained local minimum

Lagrangian

$$L(x, \lambda) = f(x) + \lambda' g(x)$$

has

$$\nabla L(x, \lambda) = \begin{bmatrix} \nabla f(x) + J_g(x)'\lambda \\ g(x) \end{bmatrix}, \qquad H_L(x, \lambda) = \begin{bmatrix} B(x, \lambda) & J_g(x)' \\ J_g(x) & O \end{bmatrix}$$

Above first-order conditions conveniently translate into (x^*, λ) being a critical point of L.

However, saddle point: H_L cannot be positive definite. Need to check $Z'B(x^*, \lambda)Z$ as discussed above.

If also inequality constraints $h_k(x) \leq 0$: corresponding $\lambda_k \geq 0$ and $h_k(x^*)\lambda_k = 0$ (k with $\lambda_k > 0$ give *active set* where $h_k(x^*) = 0$).

Sequential Quadratic Programming

Consider minimization problem with equality constraints. Suppose we use Newton's method to determine the critical points of the Lagrange function, i.e., to solve

$$\nabla L(x, \lambda) = \begin{bmatrix} \nabla f(x) + J_g(x)'\lambda \\ g(x) \end{bmatrix} = 0.$$

Gives the system

$$\begin{bmatrix} B(x_k, \lambda_k) & J_g(x_k)' \\ J_g(x_k) & O \end{bmatrix} \begin{bmatrix} s_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} \nabla f(x_k) + J_g(x_k)'\lambda_k \\ g(x_k). \end{bmatrix}$$

which are the first-order conditions for the constrained optimization problem

$$\min_s \frac{1}{2} s' B(x_k, \lambda_k) s + s'(f(x_k) + J_g(x_k)'\lambda_k), \qquad J_g(x_k)s + g(x_k) = 0.$$

Sequential Quadratic Programming

Why? Dropping arguments, want to solve

$$\frac{1}{2}s'Bs + s'u \to \min, \qquad Js + v = 0.$$

Lagrangian is $L(s, \delta) = \frac{1}{2}s'Bs + s'u + \delta'(Js + v)$ for which the first-order conditions are

$$Bs + u + J'\delta = 0, \qquad Js + v = 0$$

Equivalently,

$$\begin{bmatrix} B & J' \\ J & O \end{bmatrix} \begin{bmatrix} s \\ \delta \end{bmatrix} = - \begin{bmatrix} u \\ v \end{bmatrix}.$$

Sequential Quadratic Programming

I.e., when using Newton's method to determine the critical points of the Lagrange function, each Newton step amounts to solving a quadratic optimization problem under linear constraints: hence, method known as *Sequential Quadratic Programming*.

One does not necessarily solve the linear system directly (remember that the Hessian is symmetric but indefinite, so Choleski factorization is not possible).

Penalty Methods

Idea: convert the constrained problem into a sequence of unconstrained problems featuring a *penalty function* to achieve (approximate) feasibility.

E.g., for the equality-constrained problem

$$f(x) \to \min, \qquad g(x) = 0$$

one considers

$$\phi_\rho(x) = f(x) + \rho \|g(x)\|^2/2 \to \min;$$

Under appropriate conditions it can be shown that the solutions $x^*(\rho)$ converge to the solution x^* of the original problem for $\rho \to \infty$.

Augmented Lagrangian Methods

For penalty method, letting ρ increase without bounds often leads to problems, which motivates to instead use an *augmented Lagrangian function*

$$L_\rho(x, \lambda) = f(x) + \lambda' g(x) + \rho \|g(x)\|^2/2$$

for which the Lagrange multiplier estimates can be manipulated in ways to keep them bounded while converging to the constrained optimum: *augmented Lagrangian methods*.

Illustration where fixed $\rho > 0$ suffices: convex minimization with linear constraints.

Augmented Lagrangian Methods

Primal problem

$$f(x) \to \min, \qquad Ax = b.$$

with f convex. Duality theory: with Lagrangian $L(x, \lambda) = f(x) + \lambda'(Ax - b)$, dual function is $\inf_x L(x, \lambda) = -f^*(-A'\lambda) - b'\lambda$ (where f^* is the convex conjugate), and optimality conditions are

$$Ax - b = 0, \qquad \nabla f(x) + A'\lambda = 0$$

(primal and dual feasibility).

Augmented Lagrangian Methods

Consider augmented Lagrangian

$$L_\rho(x, \lambda) = f(x) + \lambda'(Ax - b) + (\rho/2)\|Ax - b\|^2.$$

Can be viewed as unaugmented Lagrangian for

$$f(x) + (\rho/2)\|Ax - b\|^2 \to \min, \qquad Ax = b.$$

Consider iteration

$$x_{k+1} = \operatorname{argmin}_x L_\rho(x, \lambda_k), \qquad \lambda_{k+1} = \lambda_k + \rho(Ax_{k+1} - b).$$

Augmented Lagrangian Methods

By definition, x_{k+1} minimizes $L_\rho(x, \lambda_k)$, hence

$$\begin{aligned} 0 &= \nabla_x L_\rho(x_{k+1}, \lambda_k) \\ &= \nabla_x f(x_{k+1}) + A'(\lambda_k + \rho(Ax_{k+1} - b)) \\ &= \nabla_x f(x_{k+1}) + A'\lambda_{k+1}. \end{aligned}$$

Thus, iteration ensures dual feasibility, convergence implies primal feasibility, hence altogether optimality.

General case uses the same idea, but may need increasing $\rho = \rho_k$.

(For additional inequality constraints, add slack variables and use bound constraints.)

Barrier Methods

Also known as interior point methods.

Use barrier functions which ensure feasibility and increasingly penalize feasible points as they increase the boundary of the feasible set.

E.g., with constraints $h(x) \leq 0$,

$$\phi_\mu(x) = f(x) - \mu \sum_i \frac{1}{h_i(x)}, \qquad \phi_\mu(x) = f(x) - \mu \sum_i \log(-h_i(x))$$

are the inverse and logarithmic barrier functions, respectively.

Idea: for $\mu > 0$, unconstrained minimum x^*_μ of ϕ_μ is necessarily feasible; as $\mu \to 0$, $x^*_\mu \to x^*$ (the constrained minimum).

Need starting values and keep all approximations in the interior of the feasible set, but allow convergence to points on the boundary.

Outline

- Root Finding
- Unconstrained Optimization
- Constrained Optimization
- Optimization with R

Optimization with R: Base Packages

- `nlm()` does a "Newton-type algorithm" for unconstrained minimization.

Optimization with R: Base Packages

- `nlm()` does a "Newton-type algorithm" for unconstrained minimization.
- `optim()` provides several methods, including the Nelder-Mead simplex algorithm (default: derivative-free but slow, not discussed above), quasi-Newton (BFGS), conjugate gradient as well as simulated annealing (a stochastic global optimization method, not discussed above).
  The quasi-Newton method can also handle *box constrains* of the form $l_i \leq x_i \leq u_i$ (where l_i and r_i may be $-\infty$ and ∞, respectively), known as L-BFGS-B.

Optimization with R: Base Packages

- `nlm()` does a "Newton-type algorithm" for unconstrained minimization.
- `optim()` provides several methods, including the Nelder-Mead simplex algorithm (default: derivative-free but slow, not discussed above), quasi-Newton (BFGS), conjugate gradient as well as simulated annealing (a stochastic global optimization method, not discussed above).
  The quasi-Newton method can also handle *box constrains* of the form $l_i \leq x_i \leq u_i$ (where l_i and r_i may be $-\infty$ and ∞, respectively), known as L-BFGS-B.
- `nlminb()`, now documented to be "for historical compatibility", provides code from the PORT library by Gay for unconstrained or box-constrained optimization using trust region methods.

Optimization with R: Base Packages

- `constrOptim()` performs minimization under linear inequality constraints using an adaptive (logarithmic) barrier algorithm in combination with the Nelder-Mead or BFGS minimizers. Note that this is an interior point method, so there must be interior points! I.e., linear equality constraints cannot be handled.

Optimization with R: Contributed Packages

- Several add-on packages perform linear programming.

Optimization with R: Contributed Packages

- Several add-on packages perform linear programming.
- Package quadprog performs quadratic programming (strictly convex case with linear equality and/or inequality constrains).

Optimization with R: Contributed Packages

- Several add-on packages perform linear programming.
- Package quadprog performs quadratic programming (strictly convex case with linear equality and/or inequality constrains).
- Package BB provides `BBoptim()` which itself wraps around `spg()`, which implements the spectral projected gradient method for large-scale optimization with convex constraints. Needs a function for projecting on the feasible set.

Optimization with R: Contributed Packages

- Several add-on packages perform linear programming.
- Package quadprog performs quadratic programming (strictly convex case with linear equality and/or inequality constrains).
- Package BB provides `BBoptim()` which itself wraps around `spg()`, which implements the spectral projected gradient method for large-scale optimization with convex constraints. Needs a function for projecting on the feasible set.
- Package nloptr provides an interface to NLopt (`http://ab-initio.mit.edu/nlopt`), a general purpose open source library for non-linear unconstrained and constrained optimization. One needs to consult the NLopt web page to learn about available methods (see `http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms`).

Optimization with R: Contributed Packages

- Package alabama provides `auglag()`, an Augmented Lagrangian routine for smooth constrained optimization, which conveniently does not require feasible starting values.

Example

Rosenbrock banana function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Clearly has a global minimum at $x^* = (1, 1)$.

Example

Rosenbrock banana function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Clearly has a global minimum at $x^* = (1, 1)$.

Multi-dimensional generalization (e.g., `http://en.wikipedia.org/wiki/Rosenbrock_function`): *extended Rosenbrock function*

$$\sum_{i=1}^{2n} (100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2$$

(sum of n uncoupled 2-d Rosenbrock functions).