CS 450: Numerical Anlaysis[1]

Numerical Optimization

University of Illinois at Urbana-Champaign

[1] *These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book "Scientific Computing: An Introductory Survey" by Michael T. Heath (slides).*

Numerical Optimization

- Our focus will be on *continuous* rather than *combinatorial* optimization:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}$$

- We consider linear, quadratic, and general nonlinear optimization problems:

Local Minima and Convexity

- Without knowledge of the analytical form of the function, numerical optimization methods at best achieve convergence to a *local* rather than *global* minimum:

- A set is *convex* if it includes all points on any line, while a function is (strictly) convex if its (unique) local minimum is always a global minimum:

Existence of Local Minima

- *Level sets* are all points for which f has a given value, *sublevel sets* are all points for which the value of f is less than a given value:

- If there exists a closed and bounded sublevel set in the domain of feasible points, then f has has a global minimum in that set:

Optimality Conditions

- If $\boldsymbol{x}$ is an interior point in the feasible domain and is a local minima,

$$\nabla f(\boldsymbol{x}) = \left[\frac{df}{dx_1}(\boldsymbol{x}) \quad \cdots \frac{df}{dx_n}(\boldsymbol{x}) \right]^T = \mathbf{0} :$$

- *Critical points* $\boldsymbol{x}$ satisfy $\nabla f(\boldsymbol{x}) = \mathbf{0}$ and can be minima, maxima, or saddle points:

Hessian Matrix

- To ascertain whether a critical point $\boldsymbol{x}$, for which $\nabla f(\boldsymbol{x}) = \mathbf{0}$, is a local minima, consider the *Hessian matrix*:

- If $\boldsymbol{x}^*$ is a minima of f, then $\boldsymbol{H}_f(\boldsymbol{x}^*)$ is positive semi-definite:

Optimality on Feasible Region Border

- Given an equality constraint $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$, it is no longer necessarily the case that $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$. Instead, it may be that directions in which the gradient decreases lead to points outside the feasible region:

$$\exists \boldsymbol{\lambda} \in \mathbb{R}^n, \quad -\nabla f(\boldsymbol{x}^*) = \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda}$$

- Such *constrained minima* are critical points of the Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$, so they satisfy:

$$\nabla \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}^*) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}^*) \end{bmatrix} = \mathbf{0}$$

Sensitivity and Conditioning

- The condition number of solving a nonlinear equations is $1/f'(x^*)$, however for a minimizer x^*, we have $f'(x^*) = 0$, so conditioning of optimization is inherently bad:

- To analyze worst case error, consider how far we have to move from a root $\boldsymbol{x}^*$ to perturb the function value by ϵ:

Golden Section Search

Demo: Golden Section Proportions

- Given bracket $[a, b]$ with a unique minimum (f is *unimodal* on the interval), *golden section search* considers consider points $f(x_1), f(x_2)$, $a < x_1 < x_2 < b$ and discards subinterval $[a, x_1]$ or $[x_2, b]$:

- Since one point remains in the interval, golden section search selects x_1 and x_2 so one of them can be effectively reused in the next iteration:

Newton's Method for Optimization

Demo: Newton's Method in 1D

- At each iteration, approximate function by quadratic and find minimum of quadratic function:

- The new approximate guess will be given by $x_{k+1} - x_k = -f'(x_k)/f''(x_k)$:

Successive Parabolic Interpolation

- Interpolate f with a quadratic function at each step and find its minima:

- The convergence rate of the resulting method is roughly 1.324

Safeguarded 1D Optimization

- Safeguarding can be done by bracketing via golden section search:

- Backtracking and step-size control:

General Multidimensional Optimization

Demo: *Nelder-Mead Method*

- Direct search methods by simplex (*Nelder-Mead*):

- Steepest descent: find the minimizer in the direction of the negative gradient:

Convergence of Steepest Descent

Demo: Steepest Descent

- Steepest descent converges linearly with a constant that can be arbitrarily close to 1:

- Given quadratic optimization problem $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$ where $\boldsymbol{A}$ is symmetric positive definite, the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$ satisfies

Gradient Methods with Extrapolation

- We can improve the constant in the linear rate of convergence of steepest descent by leveraging *extrapolation methods*, which consider two previous iterates (maintain *momentum* in the direction $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$):

- The *heavy ball method*, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

Conjugate Gradient Method

Demo: *Conjugate Gradient Method*

- The *conjugate gradient method* is capable of making the optimal choice of α_k and β_k at each iteration of an extrapolation method:

- *Parallel tangents* implementation of the method proceeds as follows

Krylov Optimization

Demo: *Conjugate Gradient Parallel Tangents as Krylov Subspace Method*

- Conjugate Gradient finds the minimizer of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$ within the Krylov subspace of $\boldsymbol{A}$:

Newton's Method

Demo: Newton's Method in n dimensions

- Newton's method in n dimensions is given by finding minima of n-dimensional quadratic approximation:

$$\nabla f(x) = 0$$

x_k - current guess

$$f(x) \approx \tilde{f}(x) = \tilde{f}(x_k + s_k) = f(x_k) + \nabla f(x_k) s_k + \frac{1}{2} s_k^T H_f(x_k) s_k$$

find s_k so that

$$\nabla_{s_k} f(x_k + s_k) = 0$$

solve

$$\nabla f(x_k) + H_f(x_k) s_k = 0 \Rightarrow H_f(x_k) s_k = -\nabla f(x_k)$$

Quasi-Newton Methods

- *Quasi-Newton* methods compute approximations to the Hessian at each step:

direction s_k can lead to increase if $f(x_k + \alpha s_k)$ for any $\alpha > 0$

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

B_k: approximate Hessian; α: line search

- The *BFGS* method is a secant update method, similar to Broyden's method:

- maintain symmetry of B_k
- cheap update of B_{k+1} from B_k
- easy to update B_{k+1}^{-1} from B_k^{-1}

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

$s_k = x_{k+1} - x_k$

BFGS $\Rightarrow$ $B_{k+1} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$

Nonlinear Least Squares

- An important special case of multidimensional optimization is *nonlinear least squares*, the problem of fitting a nonlinear function $f_{\boldsymbol{x}}(t)$ so that $f_{\boldsymbol{x}}(t_i) \approx y_i$:

$f_{[x_1,x_2]}(t) = x_1 \sin(x_2 t)$ so that $\begin{bmatrix} f_{[x_1,x_2]}(t_1) \\ \vdots \\ f_{[x_1,x_2]}(t_n) \end{bmatrix} \approx \begin{bmatrix} 3.2 \\ \vdots \\ 6.4 \end{bmatrix} \} y$

- We can cast nonlinear least squares as an optimization problem and solve it by Newton's method:

$r_i(x) = f_x(t_i) - y_i$

$\varphi(x) = \frac{1}{2}\|r(x)\|_2^2 = \frac{1}{2} r(x)^T r(x) \Rightarrow$ objective function

$x^* = \operatorname{argmin}_x \varphi(x)$

$\nabla \varphi(x) = J_r(x)^T r(x)$

$H_\varphi(x) = J_r(x)^T J_r(x) + \sum_{i=1} r_i(x) H_{r_i}(x)$

Newton's method $H_\varphi(x_k) s_k = -\nabla \varphi(x_k)$

Gauss-Newton Method

- The Hessian for nonlinear least squares problems has the form:

$$H_{f_k}(x) = J_r(x)^T J_r(x) + \sum_i r_i H_{r_i}(x)$$

goes to zero

- The *Gauss-Newton* method is Newton iteration with an approximate Hessian:

$$H_{f_k}(x) \approx J_r(x)^T J_r(x)$$

$$J_r(x_k)^T J_r(x_k) s_k = -\underbrace{J_r(x_k) r(x_k)}_{\nabla f_k(x_k)}$$

linear least squares

$$J_r(x_k) s_k \cong -r(x_k)$$

- The Levenberg-Marquardt method incorporates Tykhonov regularization into the linear least squares problems within the Gauss-Newton method.

Constrained Optimization Problems

- We now return to the general case of *constrained* optimization problems:

$$\min_x f(x) \quad \text{subject to} \quad g(x) = 0 \quad \text{and} \quad h(x) \leq 0$$

equality — easier than — inequality

- Generally, we will seek to reduce constrained optimization problems to a series of unconstrained optimization problems:
  - *sequential quadratic programming*: → sequence of QPs that are unconstrained
  - *penalty-based methods*: → sequence of unconstrained but more ill-conditioned problems
  - *active set methods*: → inequality → equality constraints

Sequential Quadratic Programming

- *Sequential quadratic programming* (SQP) corresponds to using Newton's method to solve the equality constrained optimality conditions, by finding critical points of the Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$,

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

→ equality constraints

- At each iteration, SQP computes $\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{\lambda}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{s}_k \\ \boldsymbol{\delta}_k \end{bmatrix}$ by solving

$$H_{\mathcal{L}}(x_k, \lambda_k) \begin{bmatrix} s_k \\ \delta_k \end{bmatrix} = -\nabla \mathcal{L}(x_k, \lambda_k)$$

$$\begin{bmatrix} B(x_k, \lambda_k) & J_g(x_k) \\ J_g^T(x_k) & 0 \end{bmatrix} \quad \text{where} \quad B(x, \lambda) = H_f(x) + \sum_i \lambda_i H_{g_i}(x)$$

Inequality Constrained Optimality Conditions

- The *Karush-Kuhn-Tucker (KKT)* conditions hold for local minima of a problem with equality and inequality constraints, the key conditions are
  - First, any minima x^* must be a feasible point, so $g(x^*) = 0$ and $h(x^*) \geq 0$.
  - We say the ith inequality constraint is *active* at a minima x^* if $h_i(x^*) = 0$.
  - The collection of equality constraints and active inequality constraints q^*, satisfies $q^*(x^*) = 0$.
  - The negative gradient of the objective function at the minima must be in the row span of the Jacobian of this collection of constraints:

    $-\nabla f(x^*) = J_{q^*}^T(x^*)\lambda^*$ where λ^* are Lagrange multipliers of constraints in q^*.

- To use SQP for an inequality constrained optimization problem, consider at each iteration an *active set* of constraints:

active set method at step k, we have x_k, λ_k

$q_k = \begin{bmatrix} g(x) \\ h_2(x) \\ h_4(x) \end{bmatrix}$ e.g. if $h_2(x) = 0$ and $h_4(x) \geq 0$ violated

↑ exactly satisfied ⇝ Lagrange multipliers corresp. to q_k

SQP step on $\mathcal{L}_k(x, \lambda) = f(x) + \lambda^T q_k(x)$

Penalty Functions

- Alternatively, we can reduce constrained optimization problems to unconstrained ones by modifying the objective function. *Penalty* functions are effective for equality constraints $\boldsymbol{g}(\boldsymbol{x}) = 0$:

$$Q_\rho(x) = f(x) + \rho \langle g(x), g(x) \rangle$$

ρ: parameter

solve sequence of unconstrained problems $\min_x Q_\rho(x)$ with $\rho \to \infty$, so that $x \to x^*$

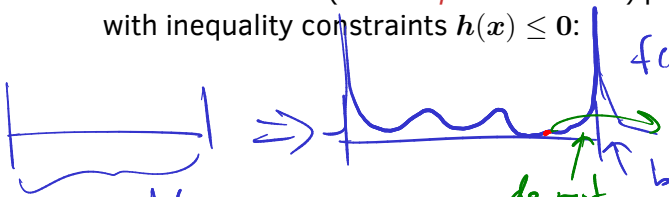
- The augmented Lagrangian function provides a more numerically robust approach:

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x) + \rho \langle g(x), g(x) \rangle$$

Barrier Functions

stay within feasible region

- *Barrier functions* (*interior point methods*) provide an effective way of working with inequality constraints $\boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$:



inverse barrier function

$$\varphi(x) = f(x) - \mu \frac{1}{h(x)}$$

$h(x) \leq 0$
so as $h(x) \to 0$
$\mu \frac{1}{h(x)} \to -\infty$

$\mu \downarrow$ take $\mu \to 0$

so that $\underset{x}{\operatorname{argmin}}\,(\varphi(x)) \to \underset{x}{\operatorname{argmin}}\,(f(x))$ s.t. $h(x) \leq 0$

logarithmic barrier function: $\varphi(x) = f(x) - \mu \log(-h(x))$