6. Unconstrained optimization

- unconstrained minimization
- descent methods
- the gradient descent method
- Newton's method

Unconstrained minimization

$$\text{minimize} \quad f(\boldsymbol{x}) \tag{6.1}$$

- $\boldsymbol{x} = (x_1, \ldots, x_n)$ is the *variable*
- $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*

Solution: the point $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_n^\star)$ is a *minimizer (minimum point)* of f or *solution* of (6.1) if

$$f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$$

for all n-vectors $\boldsymbol{x}$

Optimal value and local minimizer

Optimal value: the *optimal value* of the minimization problem is the greatest p such that $p \leq f(\boldsymbol{x})$, denoted by $\min f(\boldsymbol{x})$

- if $\boldsymbol{x}^\star$ is a minimizer of f, then $f(\boldsymbol{x}^\star) = \min f(\boldsymbol{x})$ and we say that the optimal value is attained at $\boldsymbol{x}^\star$
- if $\min f(\boldsymbol{x}) = -\infty$, then we say that the function is unbounded below
- the optimal value is unique even though there could be multiple solutions

Local minimizer

- the minimizer $\boldsymbol{x}^\star$ of f is also called a *global minimizer* of f
- a vector $\boldsymbol{x}^o$ is a *local minimizer* or *local minimum point* if there exists a scalar $r > 0$ such that $f(\boldsymbol{x}^o) \leq f(\boldsymbol{x})$ for all $\|\boldsymbol{x} - \boldsymbol{x}^o\| \leq r$; it is a *strict local minimizer* if $f(\boldsymbol{x}^o) < f(\boldsymbol{x})$

First-order necessary condition

if the n-vector $\boldsymbol{x}^o$ is a local minimizer of $f : \mathbb{R}^n \to \mathbb{R}$, then

$$\nabla f(\boldsymbol{x}^o) = \mathbf{0} \qquad \left(\frac{\partial f}{\partial x_i}(\boldsymbol{x}^o) = 0, \quad i = 1, \ldots, n\right)$$

- this condition is *necessary* but not sufficient; points that satisfy $\nabla f(\hat{\boldsymbol{x}}) = \mathbf{0}$ can be minimizers, maximizers, or neither (saddle points)
- points that satisfies $\nabla f(\hat{\boldsymbol{x}}) = \mathbf{0}$ are called *stationary points* or *critical points*
- in general, to find a global minimizer, we need to check whether the solutions of $\nabla f(\hat{\boldsymbol{x}}) = \mathbf{0}$ are in fact global minimizers
- it often is very difficult to solve the set of nonlinear equations and numerical algorithms are often used for finding stationary points

Example 6.1

find the stationary points of

$$f(\boldsymbol{x}) = x_1^3 - x_1^2 x_2 + 2x_2^2$$

setting the gradient (partial derivatives) to zero, we get the FONC:

$$\frac{\partial f}{\partial x_1} = 3x_1^2 - 2x_1 x_2 = 0$$
$$\frac{\partial f}{\partial x_2} = -x_1^2 + 4x_2 = 0$$

solving, we get two stationary points: $(0, 0)$ and $(6, 9)$

Second-order condition

Necessary condition: if $\boldsymbol{x}^o$ is a local minimizer, then $\nabla f(\boldsymbol{x}^o) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}^o) \geq 0$

Sufficient condition: if $\nabla f(\boldsymbol{x}^o) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}^o) > 0$, then $\boldsymbol{x}^o$ is a strict local minimizer

Necessary and sufficient condition: if $\nabla^2 f(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$ ('f is convex'), then $\boldsymbol{x}^\star$ is global minimizer if and only if $\nabla f(\boldsymbol{x}^\star) = \mathbf{0}$

- for single variable, the Hessian is just the second derivative $f''(x)$
- we can find maximizers by finding minimizers of $-f$

Example 6.2

find the stationary points of $f(\boldsymbol{x})$ and, if possible, determine whether they are local or global minimizers

a) $f(\boldsymbol{x}) = x_1^3 - x_1^2 x_2 + 2x_2^2$

b) $f(\boldsymbol{x}) = \frac{1}{2}x_1^2 + x_1 x_2 + 2x_2^2 - 4x_1 - 4x_2 - x_2^3$

c) $f(\boldsymbol{x}) = x_1^2 - x_1 x_2 + x_2^2 - 3x_2$

a) for $f(\boldsymbol{x}) = x_1^3 - x_1^2 x_2 + 2x_2^2$, setting the gradient, we find that the stationary points are $(0,0)$ and $(6,9)$ (see page 6.5); the Hessian is

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} 6x_1 - 2x_2 & -2x_1 \\ -2x_1 & 4 \end{bmatrix}$$

hence,

$$\nabla^2 f(0,0) = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}, \quad \nabla^2 f(6,9) = \begin{bmatrix} 18 & -12 \\ -12 & 4 \end{bmatrix}$$

- since $\nabla^2 f(0,0)$ is only positive semidefinite, it is still unclear whether $(0,0)$ is a local minimizer
- since the matrix $\nabla^2 f(6,9)$ is indefinite, the point $(6,9)$ is not a local minimizer/maximizer
- since $f(\epsilon, 0) > 0$ for any $\epsilon > 0$ and $f(\epsilon, 0) < 0$ for any $\epsilon < 0$, we conclude that the point $(0,0)$ is not a local minimizer/maximizer

b) for $f(\boldsymbol{x}) = \frac{1}{2}x_1^2 + x_1x_2 + 2x_2^2 - 4x_1 - 4x_2 - x_2^3$, the FONC is

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 + x_2 - 4 \\ x_1 + 4x_2 - 4 - 3x_2^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

solving, we get the stationary points $(4,0)$ and $(3,1)$; the Hessian is

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} 1 & 1 \\ 1 & 4 - 6x_2 \end{bmatrix}$$

thus,

$$\nabla^2 f(4,0) = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}, \quad \nabla^2 f(3,1) = \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}$$

- since $\nabla^2 f(4,0) > 0$ and $\nabla^2 f(3,1)$ is indefinite, $\hat{\boldsymbol{x}} = (4,0)$ is a local minimizer and $(3,1)$ is not a minimizer/maximizer
- note that the point $\hat{\boldsymbol{x}} = (4,0)$ is not a global minimizer since $f(0, x_2) \to -\infty$ as $x_2 \to \infty$

c) for $f(\boldsymbol{x}) = x_1^2 - x_1x_2 + x_2^2 - 3x_2$, the FONC is

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} 2x_1 - x_2 \\ -x_1 + 2x_2 - 3 \end{bmatrix} = \mathbf{0}$$

or

$$\begin{aligned} 2x_1 - x_2 &= 0 \\ -x_1 + 2x_2 &= 3 \end{aligned}$$

these two equations have a unique solution $\hat{x}_1 = 1, \hat{x}_2 = 2$, which is a candidate for a global minimizer; since the Hessian

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

is positive definite, the point $\hat{\boldsymbol{x}} = (1, 2)$ is a global minimizer

Example 6.3 (quadratic functions)

suppose we want to minimize $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{r}^T \boldsymbol{x} + c$ where Q is an $n \times n$ symmetric matrix; the FONC is

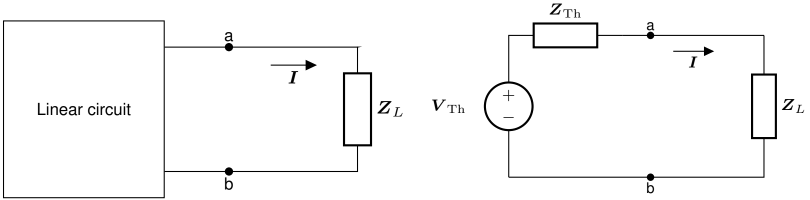
$$\nabla f(\boldsymbol{x}) = Q\boldsymbol{x} + \boldsymbol{r} = \mathbf{0}$$

the Hessian is $\nabla^2 f(\boldsymbol{x}) = Q$

- if $Q \geq 0$, then $\boldsymbol{x}^\star$ is a global minimizer iff $Q\boldsymbol{x}^\star + \boldsymbol{r} = \mathbf{0}$
  - if $\boldsymbol{r} \notin \mathrm{range}(Q)$, then there is no solution and f is unbounded below
  - if $Q > 0$, then there is a unique minimizer $\boldsymbol{x}^\star = -Q^{-1}\boldsymbol{r}$
  - if Q is singular but $\boldsymbol{r} \in \mathrm{range}(Q)$, then there exists multiple solutions
- if Q is indefinite, then any point satisfying the FONC is a saddle-point (not a minimizer/maximizer)
- if Q is invertible, then there is a unique stationary point:

$$\hat{\boldsymbol{x}} = -Q^{-1}\boldsymbol{r}$$

Example 6.4 (maximum power transfer)



- $\boldsymbol{V}_{\rm Th}$ is the Thevenin voltage, $\boldsymbol{Z}_{\rm Th} = R_{\rm Th} + jX_{\rm Th}$ ($j = \sqrt{-1}$ is the Thevenin impedance, $\boldsymbol{Z}_L = R_L + jX_L$ is the impedance of the load
- from circuit analysis, the average power delivered to the load is

$$P = |\boldsymbol{I}|^2 R_L, \qquad \boldsymbol{I} = \frac{\boldsymbol{V}_{\rm Th}}{R_{\rm Th} + R_L + j(X_{\rm Th} + X_L)}.$$

we want to find the load impedance (*i.e.*, R_L and X_L) such that average power delivered to the load P is maximized; (suppose that $\boldsymbol{V}_{\rm Th} = 1$ and $R_{\rm Th} > 0$)

we can maximize the power by solving

$$\text{maximize} \quad f(\boldsymbol{x}) = \frac{x_1}{(R_{\rm Th}+x_1)^2 + (X_{\rm Th}+x_2)^2}$$

setting the gradient (partial derivatives) to zero:

$$\nabla_{x_1} f(\boldsymbol{x}) = \frac{\partial f}{\partial x_1} = \frac{(R_{\rm Th}+x_1)^2 + (X_{\rm Th}+x_2)^2 - 2x_1(R_{\rm Th}+x_1)}{\left[(R_{\rm Th}+x_1)^2 + (X_{\rm Th}+x_2)^2\right]^2} = 0$$

$$\nabla_{x_2} f(\boldsymbol{x}) = \frac{\partial f}{\partial x_2} = \frac{-2x_1(X_{\rm Th}+x_2)}{\left[(R_{\rm Th}+x_1)^2 + (X_{\rm Th}+x_2)^2\right]^2} = 0$$

from the second equation, we have $x_1 = 0$ or $x_2 = -X_{\rm Th}$; note that $x_1 = 0$ does not satisfy the first condition; using $x_2 = -X_{\rm Th}$ into the second condition and simplifying, we get

$$(R_{\rm Th}+x_1)^2 - 2x_1(R_{\rm Th}+x_1) = 0 \iff x_1 = R_{\rm Th}$$

hence, the stationary point is

$$\boldsymbol{x} = (R_{\rm Th}, -X_{\rm Th})$$

we now check the second-order conditions; to simplify derivation of the Hessian, we let

$$f(\boldsymbol{x}) = g(A\boldsymbol{x} + \boldsymbol{b})$$

where

$$g(y_1, y_2, y_3) = \frac{y_1}{y_2^2 + y_3^2}, \quad A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ R_{\mathrm{Th}} \\ X_{\mathrm{Th}} \end{bmatrix}$$

the Hessian of f is $A^T \nabla^2 g(A\boldsymbol{x} + \boldsymbol{b})A$; thus, we need to find the Hessain of h; the gradient of g is

$$\nabla g(\boldsymbol{y}) = \begin{bmatrix} \frac{1}{y_2^2+y_3^2} \\ \frac{-2y_1y_2}{\left(y_2^2+y_3^2\right)^2} \\ \frac{-2y_1y_3}{\left(y_2^2+y_3^2\right)^2} \end{bmatrix}$$

the Hessian of g is

$$\nabla^2 g(\boldsymbol{y}) = \begin{bmatrix} 0 & \frac{-2y_2}{(y_2^2+y_3^2)^2} & \frac{-2y_3}{(y_2^2+y_3^2)^2} \\ \frac{-2y_2}{(y_2^2+y_3^2)^2} & \frac{-2y_1\left(y_2^2+y_3^2\right)+8y_1y_2^2}{\left(y_2^2+y_3^2\right)^3} & \frac{8y_1y_2y_3}{\left(y_2^2+y_3^2\right)^3} \\ \frac{-2y_3}{(y_2^2+y_3^2)^2} & \frac{8y_1y_2y_3}{\left(y_2^2+y_3^2\right)^3} & \frac{-2y_1\left(y_2^2+y_3^2\right)+8y_1y_3^2}{\left(y_2^2+y_3^2\right)^3} \end{bmatrix}$$

$$= \frac{2}{(y_2^2+y_3^2)^2} \begin{bmatrix} 0 & -y_2 & -y_3 \\ -y_2 & \frac{-y_1\left(y_2^2+y_3^2\right)+4y_1y_2^2}{\left(y_2^2+y_3^2\right)} & \frac{4y_1y_2y_3}{\left(y_2^2+y_3^2\right)} \\ -y_3 & \frac{4y_1y_2y_3}{\left(y_2^2+y_3^2\right)} & \frac{-y_1\left(y_2^2+y_3^2\right)+4y_1y_3^2}{\left(y_2^2+y_3^2\right)} \end{bmatrix}$$

note that at $\boldsymbol{x} = (R_{\mathrm{Th}}, -X_{\mathrm{Th}})$

$$A\boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} R_{\mathrm{Th}} \\ 2R_{\mathrm{Th}} \\ 0 \end{bmatrix}$$

hence, at $\boldsymbol{x} = (R_{\rm Th}, -X_{\rm Th})$, we have

$$\nabla^2 g(A\boldsymbol{x}+\boldsymbol{b}) = \frac{2}{(2R_{\rm Th})^4}\begin{bmatrix} 0 & -2R_{\rm Th} & 0 \\ -2R_{\rm Th} & 3R_{\rm Th} & 0 \\ 0 & 0 & -R_{\rm Th} \end{bmatrix}$$
$$= \frac{1}{(2R_{\rm Th})^3}\begin{bmatrix} 0 & -2 & 0 \\ -2 & 3 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

the Hessian of f at $\boldsymbol{x} = (R_{\rm Th}, -X_{\rm Th})$ is

$$\nabla^2 f(\boldsymbol{x}) = A^T \nabla^2 g(A\boldsymbol{x}+\boldsymbol{b}) A$$
$$= \frac{1}{(2R_{\rm Th})^3}\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & -2 & 0 \\ -2 & 3 & 0 \\ 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \frac{1}{(2R_{\rm Th})^3}\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

since $R_{\rm Th} > 0$, the above Hessian is negative definite, the point $\boldsymbol{x} = (R_{\rm Th}, -X_{\rm Th})$ is a local maximum; because it is the only point stationary point, it is a global maximum

Outline

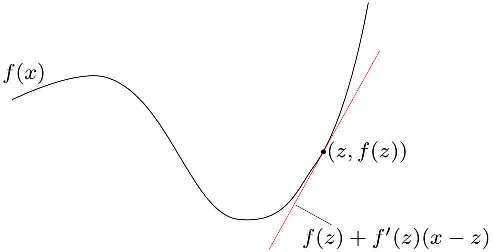
- unconstrained minimization
- **descent methods**
- the gradient descent method
- Newton's method

First-order approximation

First-order (Taylor) approximation of $f(\boldsymbol{x})$ around $\boldsymbol{z}$:

$$\begin{aligned} \hat{f}(x) &= f(z) + f'(z)(x - z) & (f : \mathbb{R} \to \mathbb{R}) \\ \hat{f}(\boldsymbol{x}) &= f(\boldsymbol{z}) + \nabla f(\boldsymbol{z})^T(\boldsymbol{x} - \boldsymbol{z}) & (f : \mathbb{R}^n \to \mathbb{R}) \\ \hat{f}(\boldsymbol{x}) &= f(\boldsymbol{z}) + Df(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{z}) & (f : \mathbb{R}^n \to \mathbb{R}^m) \end{aligned} \tag{6.2}$$

Geometrical interpretation



approximation is good if $\boldsymbol{x}$ is close to $\boldsymbol{z}$ and bad otherwise

Descent direction

Descent direction: a vector $\boldsymbol{d} \in \mathbb{R}^n$ is called a *descent direction* for f if

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) < f(\boldsymbol{x})$$

for sufficiently small $\alpha > 0$

Directional derivative: the *directional derivative* of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d}$ is

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \lim_{\alpha \to 0} \frac{f(\boldsymbol{x} + \alpha \boldsymbol{d}) - f(\boldsymbol{x})}{\alpha} = \nabla f(\boldsymbol{x})^T \boldsymbol{d} \tag{6.3}$$

- directional derivative $\nabla f(\boldsymbol{x})^T \boldsymbol{d}$ gives an approximate rate of change (increase) of f in the direction $\boldsymbol{d}$ at the point $\boldsymbol{x}$
- a vector $\boldsymbol{d} \in \mathbb{R}^n$ is a descent method if

$$f'(\boldsymbol{x}; \boldsymbol{d}) = \nabla f(\boldsymbol{x})^T \boldsymbol{d} < 0$$

Descent methods

Algorithm General descent method

choose a starting point $\boldsymbol{x}^{(0)}$, a solution tolerance $\epsilon > 0$, and a stopping criteria
repeat for $k \geq 1$

(a) determine a decent direction $\boldsymbol{d}^{(k)}$

(b) choose a stepsize α_k

(c) update $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$ if stopping criteria is satisfied, then stop and output $\boldsymbol{x}^{(k+1)}$

- α_k is called a *learning rate* or *stepsize*
- the stepsize determines the rate that $\boldsymbol{x}^{(k+1)}$ changes from $\boldsymbol{x}^{(k)}$ in the descent direction $\boldsymbol{d}^{(k)}$

Outline

- unconstrained minimization
- descent methods
- **the gradient descent method**
- Newton's method

Negative gradient direction

the directional derivative of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d} = -\nabla f(\boldsymbol{x})$ is

$$\boldsymbol{d}^T \nabla f(\boldsymbol{x}) = -\|\nabla f(\boldsymbol{x})\|^2 < 0$$

for any $\boldsymbol{x}$ with $\nabla f(\boldsymbol{x}) \neq \boldsymbol{0}$; thus, $-\nabla f(\boldsymbol{x})$ is a descent direction

- suppose $\|\boldsymbol{d}\| = 1$, then by Cauchy-Schwarz, we have

  $$-\|\nabla f(\boldsymbol{x})\| \leq \nabla f(\boldsymbol{x})^T \boldsymbol{d}$$

  and equality holds only if $\boldsymbol{d} = \nabla f(\boldsymbol{x}) / \|\nabla f(\boldsymbol{x})\|$
- hence, $-\nabla f(\boldsymbol{x})$ point in the *steepest descent* (maximum rate of decrease) direction at $\boldsymbol{x}$
- if we set $\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$ in the general descent method, we get the *gradient descent method* or *gradient descent algorithm*

Gradient descent method

Algorithm Gradient descent algorithm

given a starting point $\boldsymbol{x}^{(0)}$ and a solution tolerance $\epsilon > 0$

repeat for $k \geq 1$

1. choose a stepsize α_k
2. update

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \nabla f(\boldsymbol{x}^{(k)})$$

if $\|\nabla f(\boldsymbol{x}^{(k+1)})\| \leq \epsilon$ stop and output $\boldsymbol{x}^{(k+1)}$

- for α_k small enough, the algorithm is a descent method
- when α_k is large enough, the algorithm may not be a descent method and often does not work

Determining the stepsize

(suppose $\boldsymbol{d}^{(k)}$ is any descent direction)

Constant stepsize: set $\alpha_k = \alpha$ for all k

Exact line search

$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \quad f\big(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}\big)$$

- it is not always possible to actually find the exact minimizer α
- called *the method of steepest descent* if $\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$

Backtracking line search: choose $\beta \in (0, 1)$, and $\gamma \in (0, 1)$ and start with an initial guess α_k (*e.g.*, $\alpha_k = 1$), set $\alpha_k := \beta \alpha_k$ until

$$f(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}) - f(\boldsymbol{x}^{(k)}) < \gamma \alpha_k \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}$$

this method is a compromise between the above two methods

Stopping criteria

1. $|f(\boldsymbol{x}^{(k+1)}) - f(\boldsymbol{x}^{(k)})| < \epsilon$
2. $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| < \epsilon$
3. $|f(\boldsymbol{x}^{(k+1)}) - f(\boldsymbol{x}^{(k)})|/|f(\boldsymbol{x}^{(k)})| < \epsilon$
4. $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|/|\boldsymbol{x}^{(k)}| < \epsilon$
5. $\|\nabla f(\boldsymbol{x}^{(k)})\| < \epsilon$

- the above conditions do not necessarily imply that $\boldsymbol{x}^{(k)}$ is a good solution since it can be a local minimizer/maximizer or a saddle-point (unless f is convex)
- it is common to run the algorithm from different starting points and choose the best solution of these multiple runs

Example 6.5

$$f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$$

the gradient of this function is

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} 4(x_1 - 4)^3 \\ 2(x_2 - 3) \\ 16(x_3 + 5)^3 \end{bmatrix}$$

let the initial point be $\boldsymbol{x}^{(0)} = (4, 2, -1)$; applying one iteration of the gradient descent with $\alpha = 0.002$ gives

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 4 \\ 2 \\ -1 \end{bmatrix} - 0.002 \begin{bmatrix} 4(4-4)^3 \\ 2(2-3) \\ 16(-1+5)^3 \end{bmatrix} = \begin{bmatrix} 4.000 \\ 2.004 \\ -3.048 \end{bmatrix}$$

notice that

$$59.06 = f(4, 2.004, -3.048) < f(4, 2, -1) = 1025$$

this shows that $\alpha = 0.002$ is a good choice if we use exact line search, then

$$\begin{aligned}\alpha_0 &= \underset{\alpha>0}{\operatorname{argmin}}\, f(\boldsymbol{x}^{(0)} - \alpha\nabla f(\boldsymbol{x}^{(0)})) \\ &= \underset{\alpha>0}{\operatorname{argmin}}(0 + (2 + 2\alpha - 3)^2 + 4(-1 - 1024\alpha + 5)^4) = 3.967 \times 10^{-3}\end{aligned}$$

hence,

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} - \alpha_0 \nabla f(\boldsymbol{x}^{(0)}) = (4.000, 2.008, -5.062)$$

Example 6.6

$$f(x_1, x_2) = \frac{x_1^2}{5} + x_2^2$$

- the gradient is $\nabla f(\boldsymbol{x}) = (\frac{2}{5}x_1, 2x_2)$
- we have

$$f\big(\boldsymbol{z} - \alpha\nabla f(\boldsymbol{z})\big) = \tfrac{1}{5}(z_1 - \tfrac{2}{5}\alpha z_1)^2 + (z_2 - 2\alpha z_2)^2$$

- finding the the stepsize at iteration k in the method of steepest descent requires solving

$$\begin{aligned}\alpha &= \underset{\alpha>0}{\operatorname{argmin}}\, f\big(\boldsymbol{z} - \alpha\nabla f(\boldsymbol{z})\big) \\ &= \underset{\alpha>0}{\operatorname{argmin}} \left(\tfrac{1}{5}(z_1 - \tfrac{2}{5}\alpha z_1)^2 + (z_2 - 2\alpha z_2)^2\right)\end{aligned}$$

- setting the derivative with respect to α to zero, we get

$$-\tfrac{4}{25}z_1(z_1 - \tfrac{2}{5}\alpha z_1) - 4z_2(z_2 - 2\alpha z_2) = 0$$

- solving for α, gives

$$\alpha = \frac{\frac{4}{25}z_1^2 + 4z_2^2}{\frac{8}{125}z_1^2 + 8z_2^2} > 0$$

- hence, the method of steepest descent is

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \frac{\frac{4}{25}(x_1^{(k)})^2 + 4(x_2^{(k)})^2}{\frac{8}{125}(x_1^{(k)})^2 + 8(x_2^{(k)})^2} \begin{bmatrix} \frac{2}{5}x_1^{(k)} \\ 2x_2^{(k)} \end{bmatrix}$$

Exact line search for quadratic functions

$$f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T Q \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}$$

where Q is positive definite; gradient method with exact line search requires solving:

$$\alpha_k = \underset{\alpha > 0}{\operatorname{argmin}}\, f\big(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}\big)$$

where $\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$

Update form

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \frac{\|\nabla f(\boldsymbol{x}^{(k)})\|^2}{\nabla f(\boldsymbol{x}^{(k)})^T Q \nabla f(\boldsymbol{x}^{(k)})} \nabla f(\boldsymbol{x}^{(k)})$$

where $\nabla f(\boldsymbol{x}) = Q\boldsymbol{x} - \boldsymbol{b}$

Derivation

- letting $\boldsymbol{d} = \boldsymbol{d}^{(k)}$ and using the chain rule, we have

$$\begin{aligned} g'(\alpha) &= \boldsymbol{d}^T \nabla f\big(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}\big) \\ &= \boldsymbol{d}^T \big(Q\big(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}\big) - \boldsymbol{b}\big) \\ &= \alpha \boldsymbol{d}^T Q \boldsymbol{d} + \boldsymbol{d}^T \big(Q\boldsymbol{x}^{(k)} - \boldsymbol{b}\big) \\ &= \alpha \boldsymbol{d}^T Q \boldsymbol{d} + \boldsymbol{d}^T \boldsymbol{d} \end{aligned}$$

- setting to zero and solving for α, we get

$$\alpha_k = \frac{\boldsymbol{d}^T \boldsymbol{d}}{\boldsymbol{d}^T Q \boldsymbol{d}}$$

where $\boldsymbol{d} = -\nabla f(\boldsymbol{x}^{(k)}) = -(Q\boldsymbol{x}^{(k)} - \boldsymbol{b})$

Convergence discussion

under certain mild assumptions, the iterates $\{\boldsymbol{x}^{(k)}\}$ of the gradient algorithm can be shown to converge to a stationary point, *i.e.*,

$$\lim_{k\to\infty} \nabla f(\boldsymbol{x}^{(k)}) = \mathbf{0}$$

- if f is convex (*e.g.*, $\nabla^2 f(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$), then the iterates $\{\boldsymbol{x}^{(k)}\}$ of gradient algorithm converges to a global minimizer

- the rate of convergence is sublinear (slow) in general and linear if $\mu I \leq \nabla^2 f(\boldsymbol{x})$ for all $\boldsymbol{x}$ and some constant $\mu > 0$

Outline

- unconstrained minimization
- descent methods
- the gradient descent method
- **Newton's method**

Newton's method

consider n nonlinear equation in n variables

$$r_1(\boldsymbol{x}) = 0, \quad r_2(\boldsymbol{x}) = 0, \quad \ldots, \quad r_n(\boldsymbol{x}) = 0$$

where $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$; we let $r(\boldsymbol{x}) = (r_1(\boldsymbol{x}), \ldots, r_n(\boldsymbol{x}))$

Newton's method: choose $\boldsymbol{x}^{(0)}$ and repeat for $k \geq 0$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - Dr(\boldsymbol{x}^{(k)})^{-1} r(\boldsymbol{x}^{(k)})$$

assumes $Dr(\boldsymbol{x}^{(k)})$ exists and nonsingular

Unconstrained optimization: if $\boldsymbol{r}(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$, we get

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \nabla^2 f(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})$$

Interpretation of Newton update

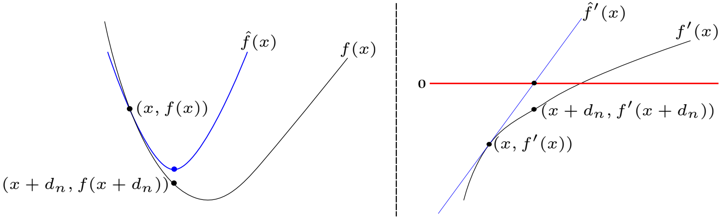
$$\boldsymbol{x} = \boldsymbol{x}^{(k)} - \nabla^2 f(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})$$

(I) minimizing the quadratic approximation of f around $\boldsymbol{x}^{(k)}$:

$$\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x}^{(k)}) + \nabla f(\boldsymbol{x}^{(k)})^T (\boldsymbol{x} - \boldsymbol{x}^{(k)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(k)})^T \nabla^2 f(\boldsymbol{x}^{(k)})(\boldsymbol{x} - \boldsymbol{x}^{(k)})$$

(II) solve approximate optimality condition around $\boldsymbol{x}^{(k)}$:

$$\widehat{\nabla f}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}^{(k)}) + \nabla^2 f(\boldsymbol{x}^{(k)})(\boldsymbol{x} - \boldsymbol{x}^{(k)}) = \mathbf{0}$$



Damped Newton's method

Algorithm Damped Newton method

given a starting point $\boldsymbol{x}^{(0)}$, a solution tolerance $\epsilon > 0$

repeat for $k \geq 1$

1. choose a stepsize α_k
2. update

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \big(\nabla^2 f(\boldsymbol{x}^{(k)})\big)^{-1} \nabla f(\boldsymbol{x}^{(k)})$$

if $\|\nabla f(\boldsymbol{x}^{(k+1)})\| \leq \epsilon$ stop and output $\boldsymbol{x}^{(k+1)}$

- assumes $\nabla^2 f(\boldsymbol{x})$ exists and is invertible
- $\boldsymbol{d}_n = -\big(\nabla^2 f(\boldsymbol{x}^{(k)})\big)^{-1} \nabla f(\boldsymbol{x}^{(k)})$ is called *Newton step* at $\boldsymbol{x}^{(k)}$
- similar stepsize selection and stopping criteria as before can be used

single-variable update

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{f'(x^{(k)})}{f''(x^{(k)})}$$

Convergence discussion

- under certain assumptions one can prove local quadratic rate of convergence (fast), *i.e.*, near the optimal solution the error $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star\|$ (where $\boldsymbol{x}^\star$ is an optimal solution) satisfies

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^\star\| \le c\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star\|^2$$

  for some positive $c > 0$

- if $\nabla^2 f(\boldsymbol{x}) > 0$ then $\boldsymbol{d}_n = -\big(\nabla^2 f(\boldsymbol{x}^{(k)})\big)^{-1}\nabla f(\boldsymbol{x}^{(k)})$ is a descent direction and quadratic convergence to the global minimizer is guaranteed under certain conditions

- does not work well if $\nabla^2 f(\boldsymbol{x})$ is not positive-definite since it is not a descent method in this case

- can use hybrid gradient-Newton method by setting $\boldsymbol{d}^{(k)} = -\nabla^2 f(\boldsymbol{x}^{(k)})^{-1}\nabla f(\boldsymbol{x}^{(k)})$ if $\nabla^2 f(\boldsymbol{x}^{(k)})$ is positive-definite and $\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$ otherwise

Numerical example I

$$f(\boldsymbol{x}) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}$$

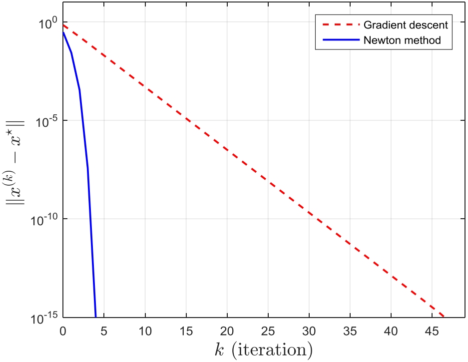
the gradient and Hessian are

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} - e^{-x_1-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} \end{bmatrix}$$

and

$$\nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1} & e^{x_1+x_2-1} - e^{x_1-x_2-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} & e^{x_1+x_2-1} + e^{x_1-x_2-1} \end{bmatrix}$$

we apply gradient descent and Newton method with initial starting point $x^{(0)} = (-1, 1)$ and step-size $\alpha = 1$



- both algorithms converge to $\boldsymbol{x}^\star = (-0.34657, 0)$
- Newton method is much faster since it uses second-order information

Matlab implementation

```
g=@(x)[exp(x(1)+x(2)-1)+exp(x(1)-x(2)-1)-exp(-x(1)-1);...
exp(x(1)+x(2)-1)-exp(x(1)-x(2)-1)]; % gradient
hess=@(x)[exp(x(1)+x(2)-1)+exp(x(1)-x(2)-1)+exp(-x(1)-1) ...
exp(x(1)+x(2)-1)-exp(x(1)-x(2)-1);...
exp(x(1)+x(2)-1)-exp(x(1)-x(2)-1) ...
exp(x(1)+x(2)-1)+exp(x(1)-x(2)-1)] % hessain
%% Newton and GD iterations
x = [-1; 1];%GD initilization
xn = [-1; 1];%Newton initilization
alpha=1; %step-size
for k=1:50
%%%Gradient descent update%%%%
grad=g(x);
if (norm(grad) < 1e-16), break; end;
x = x - alpha*grad;
%%%Newton update%%%
dn=-hess(xn)\g(xn);
xn = xn + alpha*dn;
end
```

Alternative way to construct gradient and Hessian

$$f(\boldsymbol{x}) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}$$

we can write f as $f(\boldsymbol{x}) = g(A\boldsymbol{x}+\boldsymbol{b})$, where $g(\boldsymbol{y}) = e^{y_1} + e^{y_2} + e^{y_3}$, and

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

the gradient and Hessian of g are

$$\nabla g(\boldsymbol{y}) = \begin{bmatrix} e^{y_1} \\ e^{y_2} \\ e^{y_3} \end{bmatrix}, \quad \nabla^2 g(\boldsymbol{y}) = \begin{bmatrix} e^{y_1} & 0 & 0 \\ 0 & e^{y_2} & 0 \\ 0 & 0 & e^{y_3} \end{bmatrix}$$

it follows that

$$\nabla f(\boldsymbol{x}) = A^T \nabla g(A\boldsymbol{x}+\boldsymbol{b})$$
$$\nabla^2 f(\boldsymbol{x}) = A^T \nabla^2 g(A\boldsymbol{x}+\boldsymbol{b}) A$$

Matlab implementation

```
A=[1 1;1 -1;-1 0];
b=[1;1;1];

for k=1:50
%%% Gradient descent update %%%
v=exp(A*x-b);
grad=A'*v;
if (norm(grad) < 1e-16), break; end;
x = x - alpha*grad;
%%% Newton's update %%%
vn=exp(A*xn-b);
gradn=A'*vn;
D = diag(vn);
H=A'*D*A;
dn=-H\gradn;
xn = xn + alpha*dn;
end;
```

Numerical example II

$$\text{minimize} \quad f(\boldsymbol{x}) = \sum_{i=1}^{m} \log(e^{\boldsymbol{a}_i^T \boldsymbol{x} - b_i} + e^{-\boldsymbol{a}_i^T \boldsymbol{x} + b_i})$$

- $\boldsymbol{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ are the problem data
- m and n can be very large
- suppose that we want to solve this problem using Newton's method with initialization $\boldsymbol{x}^{(0)} = \mathbf{1}$, stopping criteria $\|\nabla f(\boldsymbol{x}^{(k)})\| < 10^{-5}$, and line search parameters: $\alpha_0 = 1$, $\beta = 1/2$, and $\gamma = 0.01$
- to implement the algorithm, we first need to find the gradient and Hessian of the function f

the function f can be written as

$$f(\boldsymbol{x}) = g(A\boldsymbol{x} - \boldsymbol{b}) \quad \text{where} \quad g(\boldsymbol{y}) = \sum_{i=1}^{m} \log(e^{y_i} + e^{-y_i})$$

and

$$A = \begin{bmatrix} \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

the gradient and Hessian of h are:

$$\nabla g(\boldsymbol{y}) = \begin{bmatrix} (e^{y_1} - e^{-y_1})/(e^{y_1} + e^{-y_1}) \\ \vdots \\ (e^{y_m} - e^{-y_m})/(e^{y_m} + e^{-y_m}) \end{bmatrix}$$

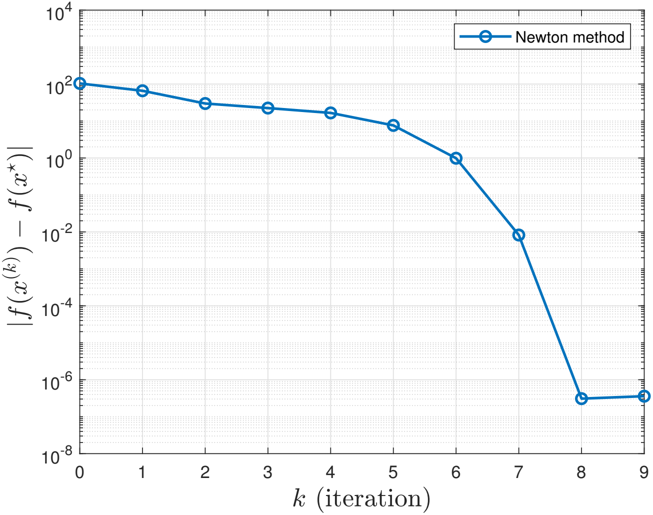
$$\nabla^2 g(\boldsymbol{y}) = \operatorname{diag}\big(4/(e^{y_1} + e^{-y_1})^2, \ldots, 4/(e^{y_m} + e^{-y_m})^2\big)$$

using the composition with affine function property, we have

$$\nabla f(\boldsymbol{x}) = A^T \nabla g(A\boldsymbol{x} - \boldsymbol{b}), \quad \nabla^2 f(\boldsymbol{x}) = A^T \nabla^2 g(A\boldsymbol{x} - \boldsymbol{b}) A$$

MATLAB code

```
alpha_0=1;
beta=0.5;
gamma=0.01;
x = ones(n,1); %initialization
k=1;
y = A*x-b;
grad = A'*((exp(y)-exp(-y))./(exp(y)+exp(-y)));
while (norm(grad) >= 1e-5)
k=k+1; %iteration counter
hess = 4*A'*diag(1./(exp(y)+exp(-y)).^2)*A;
d = -hess\grad;
alpha = alpha_0;
f = sum(log(exp(y)+exp(-y)));
while (sum(log(exp(A*(x+alpha*d)-b)+exp(-A*(x+alpha*d)+b))) ...
> f + gamma*alpha*grad'*d)
alpha = beta*alpha;
end
x = x+alpha*d;
y = A*x-b;
f = sum(log(exp(y)+exp(-y)));
grad = A'*((exp(y)-exp(-y))./(exp(y)+exp(-y)));
end
```



Newton method
10^4
10^2
10^0
10^{-2}
10^{-4}
10^{-6}
10^{-8}
$|f(x^{(k)}) - f(x^\star)|$
0
1
2
3
4
5
6
7
8
9
k (iteration)

References and further readings

- A. Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB,* SIAM, 2014.
- E. KP. Chong and S. H. Zak. *An Introduction to Optimization,* John Wiley & Sons, 2013.
- L. Vandenberghe. *EE133A lecture notes,* Univ. of California, Los Angeles. (`http://www.seas.ucla.edu/~vandenbe/ee133a.html`)
- Uri M. Ascher. *A First Course on Numerical Methods*. Society for Industrial and Applied Mathematics, 2011.