Lecture 7:

Smooth Unconstrained Minimization Algorithms

- terminology
- general descent method
- line search types
- gradient method
- steepest descent method
- Newton's method

Terminology

unconstrained minimization problem

$$\text{minimize } f(x)$$

$f : \mathbf{R}^n \to \mathbf{R}$, smooth, convex, with $\mathbf{dom}\, f = \mathbf{R}^n$

minimizing sequence: $x^{(k)}$, $k \to \infty$

$$f(x^{(k)}) \to f^\star$$

optimality condition

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

set of nonlinear equations; usually no analytical solution

more generally, if $\nabla^2 f(x) \succeq mI$, then

$$f(x) - f^\star \leq \frac{1}{2m} \|\nabla f(x)\|^2$$

. . . yields stopping criterion (if you know m)

Examples

unconstrained quadratic minimization

$$\text{minimize } x^T P x + 2q^T x + b$$

$(P = P^T \succeq 0)$

unconstrained geometric programming

$$\text{minimize } \log \sum_{i=1}^{m} e^{a_i^T x + b_i}$$

logarithmic barrier for linear inequalities

$$\begin{array}{ll} \text{minimize} & -\sum_i \log(a_i^T x + b_i) \\ \text{subject to} & a_i^T x + b_i > 0, \quad i = 1, \ldots, m \end{array}$$

is *'effectively'* unconstrained

Descent method

given starting point x
repeat
1. *Compute a search direction* v
2. *Line search.* Choose step size $t > 0$
3. *Update.* $x := x + tv$

until stopping criterion is satisfied

descent method: $f(x^{(k+1)}) < f(x^{(k)})$

(if f (quasi-)convex) v must be a **descent direction:**

$$\nabla f(x^{(k)})^T v^{(k)} < 0$$

examples

- $v^{(k)} = -\nabla f(x^{(k)})$
- $v^{(k)} = -H^{(k)} \nabla f(x^{(k)})$, $H^{(k)} = H^{(k)T} \succ 0$
- $v^{(k)} = -\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)})$

Line search types

fixed step size

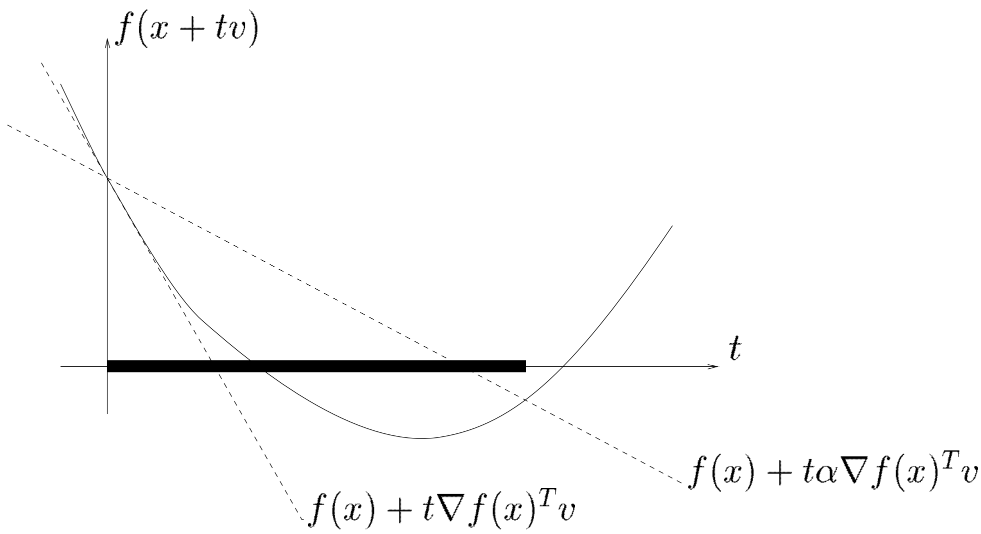
$$t = t_{\text{fixed}} > 0$$

exact line search

$$t = \operatorname*{argmin}_{s>0} f(x + sv)$$

backtracking line search $(0 < \beta < 1,\ 0 < \alpha < 0.5)$

- starting with $t = 1$, $t := \beta t$
- until $f(x + tv) \leq f(x) + t\alpha \nabla f(x)^T v$



Gradient method

given starting point x
repeat
1. *Compute search direction* $v = -\nabla f(x)$
2. *Line search.* Choose step size t
3. *Update.* $x := x + tv$

until stopping criterion is satisfied

- converges with exact or bactracking line search
- can be very slow
- almost never used in practice

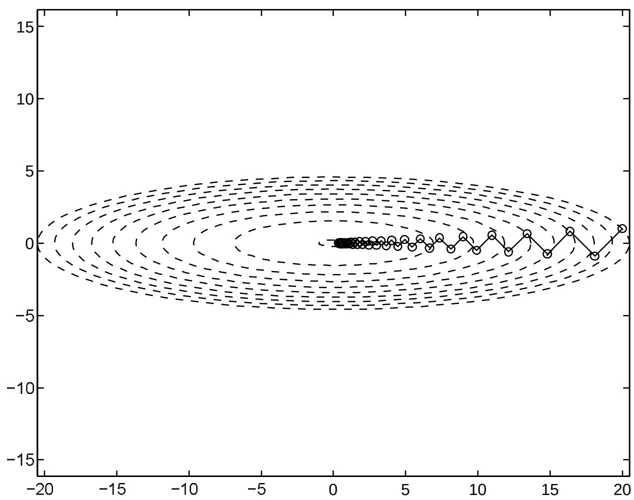
Quadratic example

$$\text{minimize } \frac{1}{2}(x^2 + My^2)$$

- exact line search
- start at $x^{(0)} = M$, $y^{(0)} = 1$ (to simplify formulas)

iterates are then

$$x^{(k)} = M\left(\frac{M-1}{M+1}\right)^k, \quad y^{(k)} = \left(-\frac{M-1}{M+1}\right)^k,$$



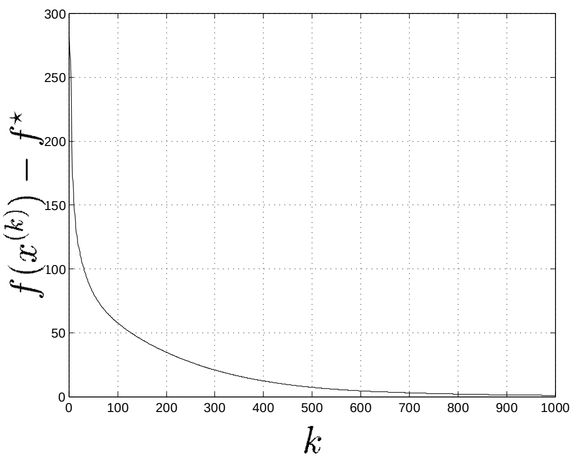
- fast if M close to 1
- slow, zig-zagging if $M \gg 1$ or $M \ll 1$

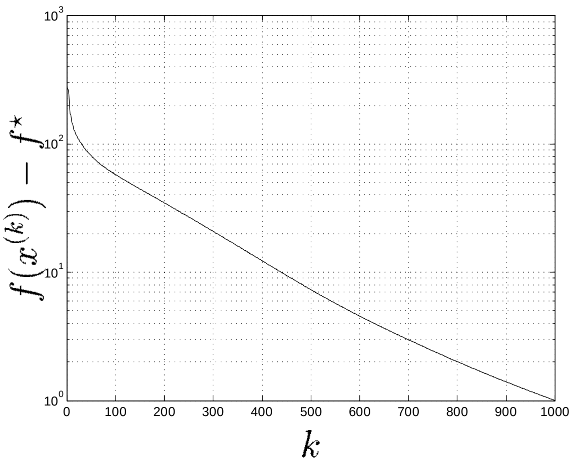
Numerical example: gradient method

$$\begin{array}{ll} \text{minimize} & c^T x - \sum_{i=1}^m \log(a_i^T x + b_i) \\ \text{subject to} & a_i^T x + b_i > 0, \quad i = 1, \ldots, m \end{array}$$

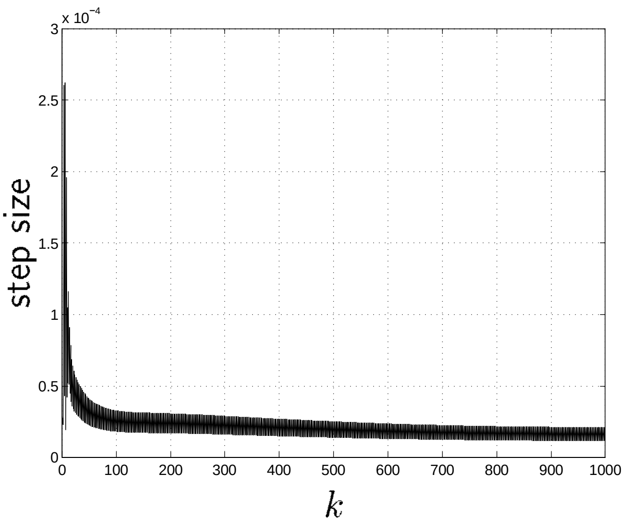
$m = 100$, $n = 50$

gradient method with exact line search





- slow convergence; zig-zagging



Steepest descent direction

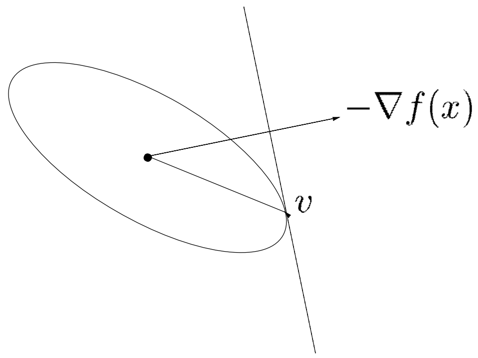
steepest descent direction for norm $\|\cdot\|_g$:

$$v = \operatorname{argmin}\{\nabla f(x)^T v \mid \|v\|_g = 1\}$$

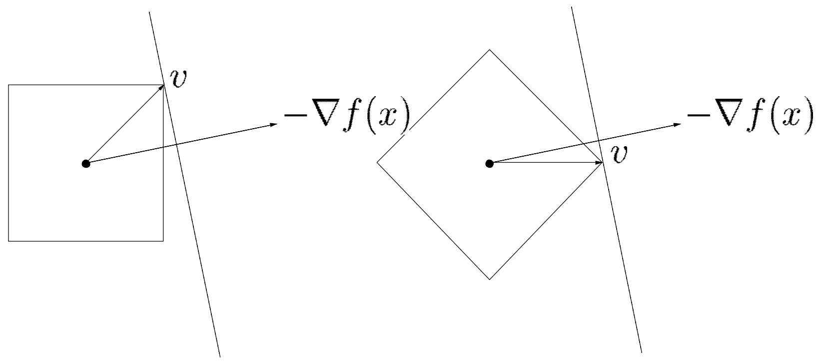
motivation: first-order approximation of f at x:

$$f(x + tv) \simeq f(x) + t\nabla f(x)^T v$$

- Euclidean norm: $v = -\nabla f(x)/\|\nabla f(x)\|$
- quadratic norm



- ℓ^∞- and ℓ^1-norm



Steepest descent method

given starting point x
repeat
1. *Compute steepest descent direction*
   $v = \mathrm{argmin}\{\nabla f(x)^T v \mid \|v\|_g = 1\}$
2. *Line search.* Choose a step size t
3. *Update.* $x := x + tv$

until stopping criterion is satisfied

- converges with exact or backtracking line search
- can be very slow

Pure Newton method

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

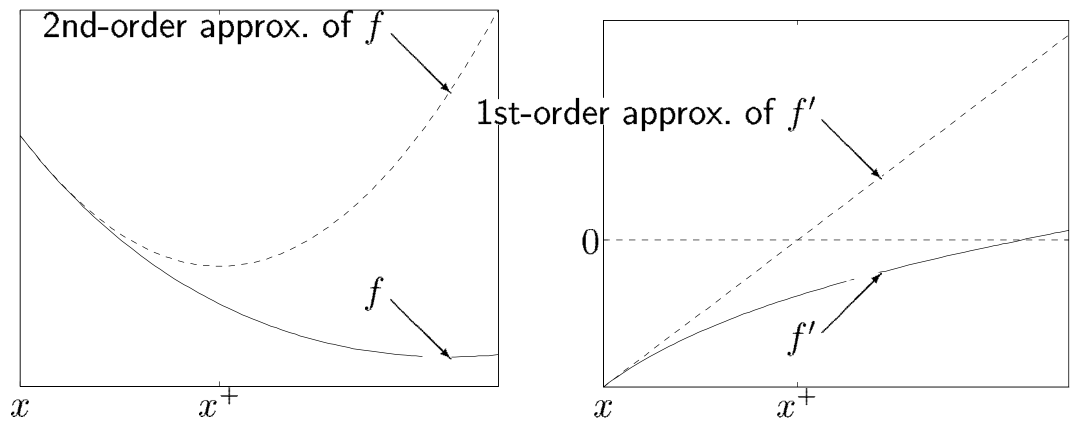
interpretations: $y = x^+$

- minimizes 2nd order expansion of f at x:

$$f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$$

- solves linearized optimality condition:

$$0 = \nabla f(x) + \nabla^2 f(x)(y - x)$$



works **very well** near optimum

Local convergence

assumptions

- $\nabla^2 f(x) \succeq mI$
- Hessian satisfies Lipschitz condition:

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|$$

(L small means f nearly quadratic)

result (see references)

$$\frac{L}{2m^2}\|\nabla f(x^+)\| \leq \left(\frac{L}{2m^2}\|\nabla f(x)\|\right)^2$$

- $\|\nabla f(x)\|$ (hence, $f(x) - f^\star$) decreases very rapidly if

$$\|\nabla f(x^{(0)})\| < \frac{m^2}{L}$$

(region of **quadratic convergence**)

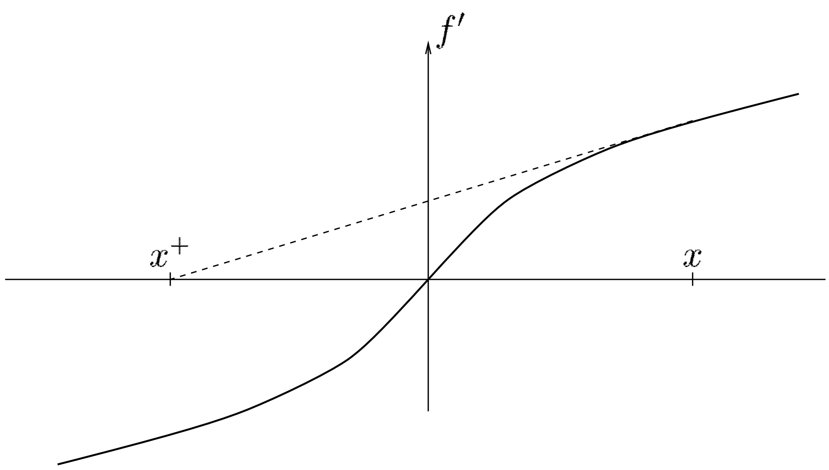
- bound on #iterations for accuracy $f(x) - f^\star \leq \epsilon$:

$$\log_2 \log_2(\epsilon_0/\epsilon), \quad \epsilon_0 = m^3/L^2$$

- practical rule of thumb: 5–6 iterations

Global behavior

pure Newton method can diverge



Damped Newton method

given starting point x
repeat
1. *Compute Newton direction* $v = -\nabla^2 f(x)^{-1} \nabla f(x)$
2. *Line search.* Choose a step size t
3. *Update.* $x := x + tv$

until stopping criterion is satisfied

- globally convergent with backtracking or exact line search
- quadratic local convergence
- hence, stopping criterion not an issue ($e.g.$, knowledge of m)

affinely invariant:

- use new coords $x = T\bar{x}$, $\det T \neq 0$
- apply Newton to $g(\bar{x}) = f(T\bar{x})$
- then $x^{(k)} = T\bar{x}^{(k)}$

$e.g.$, Newton method not affected by variable scaling (cf. gradient, steepest descent)

Convergence analysis

assumptions:

- $mI \preceq \nabla^2 f(x) \preceq MI$
- $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|$

results: (see references)

two phases:

1. damped Newton phase: $\|\nabla f(x)\| \geq \eta_1$:

$$f(x^+) \leq f(x) - \eta_2,$$

hence

$$\#\text{iterations} \leq \frac{f(x^{(0)}) - f^\star}{\eta_2}$$

2. quadratically convergent phase: $\|\nabla f(x)\| < \eta_1$

$$\#\text{iterations} \leq \log_2 \log_2(\epsilon_0/\epsilon)$$

total #iterations bounded by

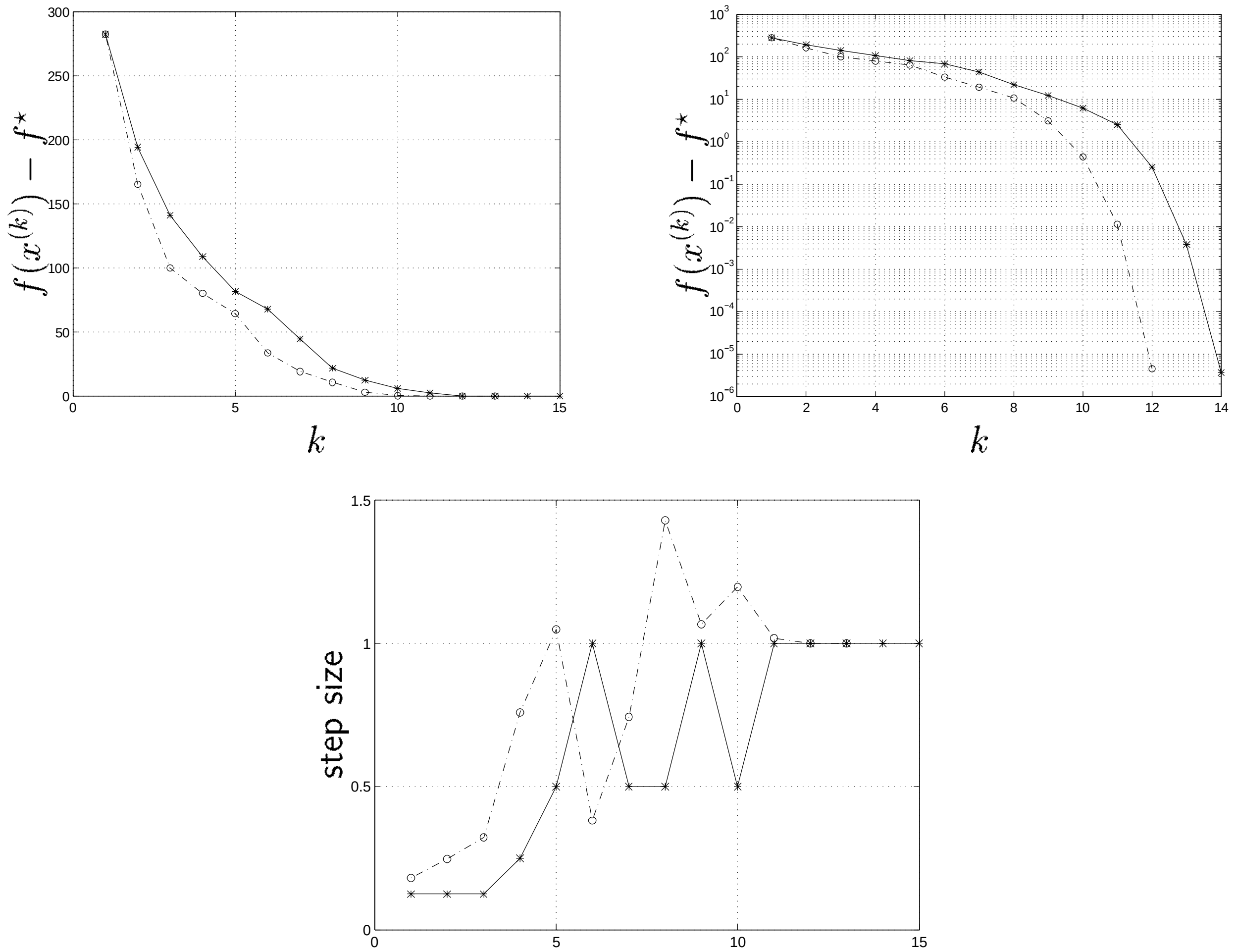
$$\frac{f(x^{(0)}) - f^\star}{\eta_2} + \log_2 \log_2(\epsilon_0/\epsilon)$$

η_1, η_2, ϵ_0 depend on m, M, L (and α, β for backtracking)

Numerical example: Newton method

$$\begin{array}{ll} \text{minimize} & c^T x - \sum_{i=1}^{m} \log(a_i^T x + b_i) \\ \text{subject to} & a_i^T x + b_i > 0, \quad i = 1, \ldots, m \end{array}$$

$m = 100$, $n = 50$



solid line: backtracking ($\beta = 0.5$, $\alpha = 0.2$)
dashed line: exact line search
(remember: each iter is more expensive than gradient method)