Lecture 15

Smooth unconstrained minimization (part one)

- terminology
- general descent method
- gradient & steepest descent methods
- Newton's method (intro)

Terminology

unconstrained minimization problem

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

$f : \mathbf{R}^n \to \mathbf{R}$, convex, differentiable
(hence **dom** f is open . . .)

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

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

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

unconstrained geometric programming

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

analytic center of linear inequalities

$$\text{minimize } -\sum_i \log(b_i - a_i^T x)$$

(**dom** $f = \{x | a_i^T x < b_i,\ i = 1, \ldots, m\}$)

Descent method

given starting point $x \in$ **dom** f
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: $f(x^{(k+1)}) < f(x^{(k)})$

since f convex, v must be a **descent direction:**

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

examples

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

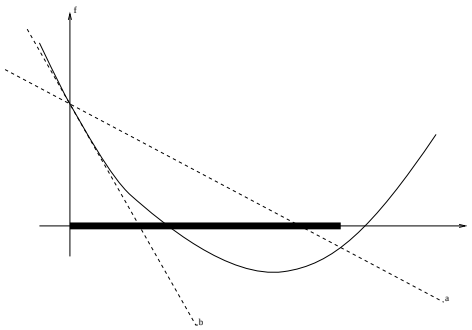
Line search types

two simple & effective line search types:

exact line search: $t = \mathbf{argmin}_{s>0}\, f(x + sv)$

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 $x \in$ **dom** f
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
- rarely used in practice

Example

$$\text{minimize } \frac{1}{2}(x_1^2 + Mx_2^2)$$

where $M > 0$

optimal point is $x^\star = 0$

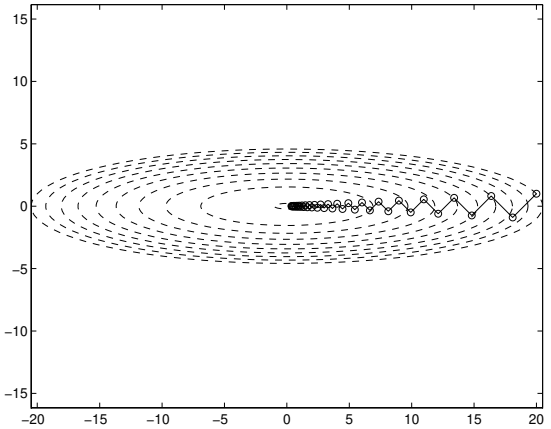
- use exact line search
- start at $x^{(0)} = (M, 1)$ (to simplify formulas)

iterates are then

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

convergence is

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



in terms of condition number $\kappa = \max\{M, 1/M\}$, convergence rate is

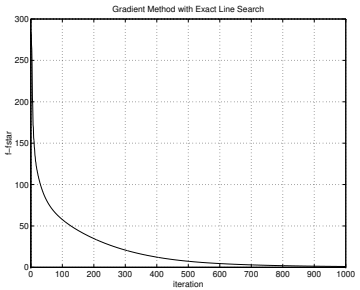
$$\left(\frac{1 - 1/\kappa}{1 + 1/\kappa}\right)^k$$

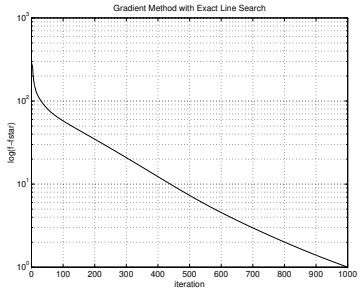
Numerical example: gradient method

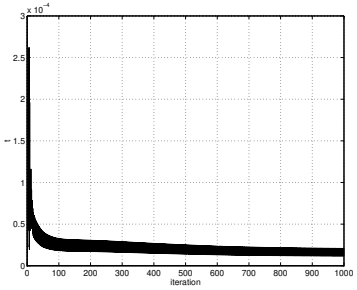
$$\text{minimize } c^T x - \sum_{i=1}^{m} \log(a_i^T x + b_i)$$

$m = 100$, $n = 50$

gradient method with exact line search







slow convergence; zig-zagging

Steepest descent direction

first-order approximation of f at x:

$$f(x+z) \approx f(x) + \nabla f(x)^T z$$

$\nabla f(x)^T z$ gives approximate decrease in f for (small) step z

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

$$v_{\mathrm{sd}} = \textbf{argmin}\{\nabla f(x)^T v \mid \|v\| = 1\}$$

gives greatest (approximate) decrease in f, per length of step (measured by $\|\cdot\|$)

Euclidean norm: $v_{\mathrm{sd}} = -\nabla f(x)/\|\nabla f(x)\|$

quadratic norm: $\|z\|_P = \left(z^T P z\right)^{1/2}$, $P = P^T \succ 0$

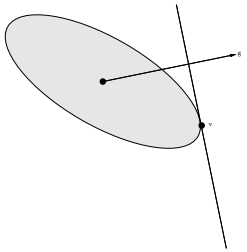
$$v_{\mathrm{sd}} = -\left(\nabla f(x)^T P^{-1} \nabla f(x)\right)^{-1/2} P^{-1/2} \nabla f(x)$$

can express v_{sd} as

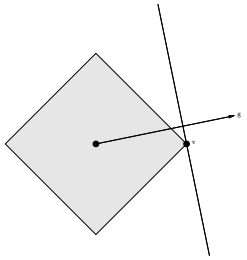
$$v_{\mathrm{sd}} = \textbf{argmin}\{\nabla f(x)^T v \mid \|v\| \leq 1\}$$

geometric interpretation: go as far as possible in direction $-\nabla f(x)$, while staying in unit ball

quadratic norm:



ℓ_1-norm:



Steepest descent method

given starting point $x \in \textbf{dom}\, f$
repeat
1. *Compute steepest descent direction* $v_{\mathrm{sd}} = \textbf{argmin}\{\nabla f(x)^T v \mid \|v\| = 1\}$
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
- sometimes v_{sd} is scaled between 1 and 2
- can be very slow
- used in special cases where v is cheap to compute