Convex Optimization

First order methods

Today

- Large scale problems: complexity
- First-order methods

Large scale problems

- Some problems coming from statistics, biology scheduling etc may have more than 10^6 variables
- A matrix of dimension 10^4 requires 800Mb of memory in double precision
- Also: a high target precision is not always necessary

First-order methods

Subgradient

- Suppose that f is a convex function with $\mathbf{dom} f = \mathbf{R}^n$, and that there is a vector $g \in \mathbf{R}^n$ such that:

$$f(y) \geq f(x) + g^T(y - x), \quad \text{for all } y \in \mathbf{R}^n$$

- The vector g is called a **subgradient** of f at x
- Of course, if f is differentiable, the gradient of f at x satisfies this condition
- The subgradient defines a **supporting hyperplane** for f at the point x

Subgradient Methods

Subgradient method:

- Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is convex
- We update the current point x_k according to:

$$x_{k+1} = x_k + \alpha_k g_k$$

  where g_k is a subgradient of f at x_k
- α_k is the step size sequence
- Similar to gradient descent but, not a descent method . . .
- Instead: use the best point and the minimum function value found so far

Subgradient Methods

Step size strategies:

- Constant step size: $\alpha_k = h$ for all $k \geq 0$
- Constant step length: $\alpha_k / \|g_k\| = h$ for all $k \geq 0$
- Square summable but not summable:

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

- Nonsummable diminishing:

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \lim_{k \to \infty} \alpha_k = 0$$

Subgradient Methods

Convergence:

Assuming $\|g\|_2 \leq G$, for all $g \in \partial f$, we can show

$$f_{\text{best}} - f^\star \leq \frac{\textbf{dist}(x_1, x^*) + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i}$$

For constant step $\alpha_i = h$, this becomes

$$f_{\text{best}} - f^\star \leq \frac{\textbf{dist}(x_1, x^*)}{2hk} + G^2 h/2$$

to get an ϵ solution, we set $h = 2\epsilon/G^2$ and

$$\frac{\textbf{dist}(x_1, x^*)}{2hk} \leq \epsilon$$

hence

$$k \geq \frac{\textbf{dist}(x_1, x^*) G^2}{4\epsilon^2}.$$

Subgradient Methods

- If the problem has constraints:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

  where $C \subset \mathbf{R}^n$ is a convex set

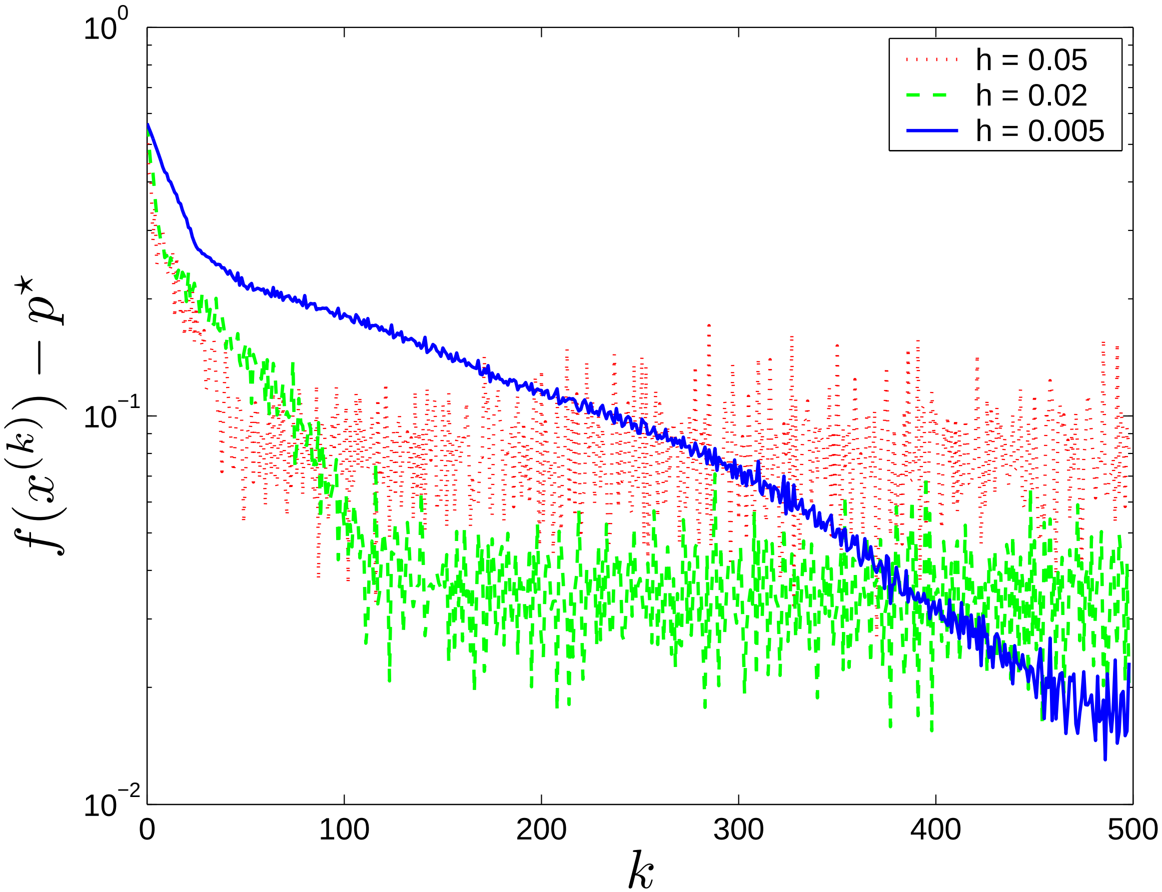
- Use the Euclidean projection $p_C(g_k)$ of the subgradient g_k on C

$$x_{k+1} = x_k + \alpha_k p_C(g_k)$$

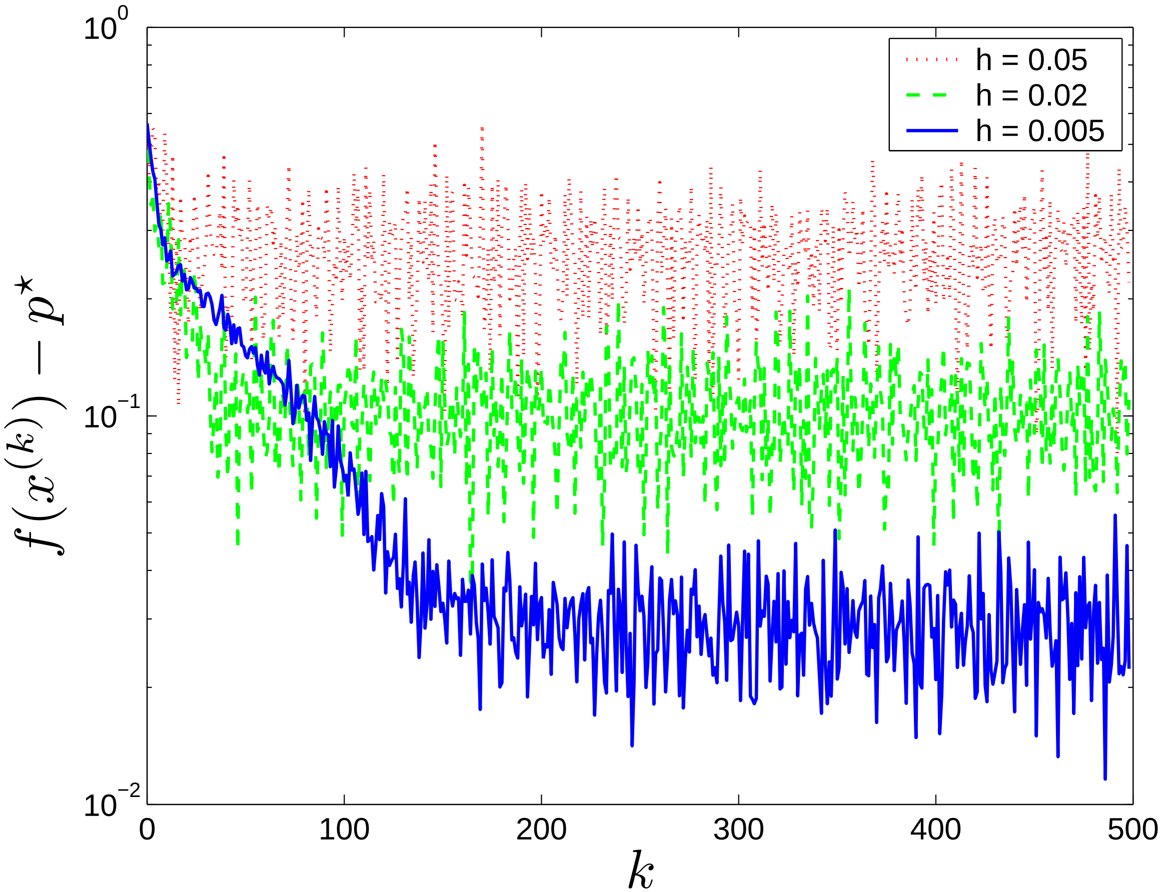
- Some numerical examples on piecewise linear minimization. . . Problem instance with $n = 10$ variables, $m = 100$ terms

Subgradient Methods: Numerical Examples

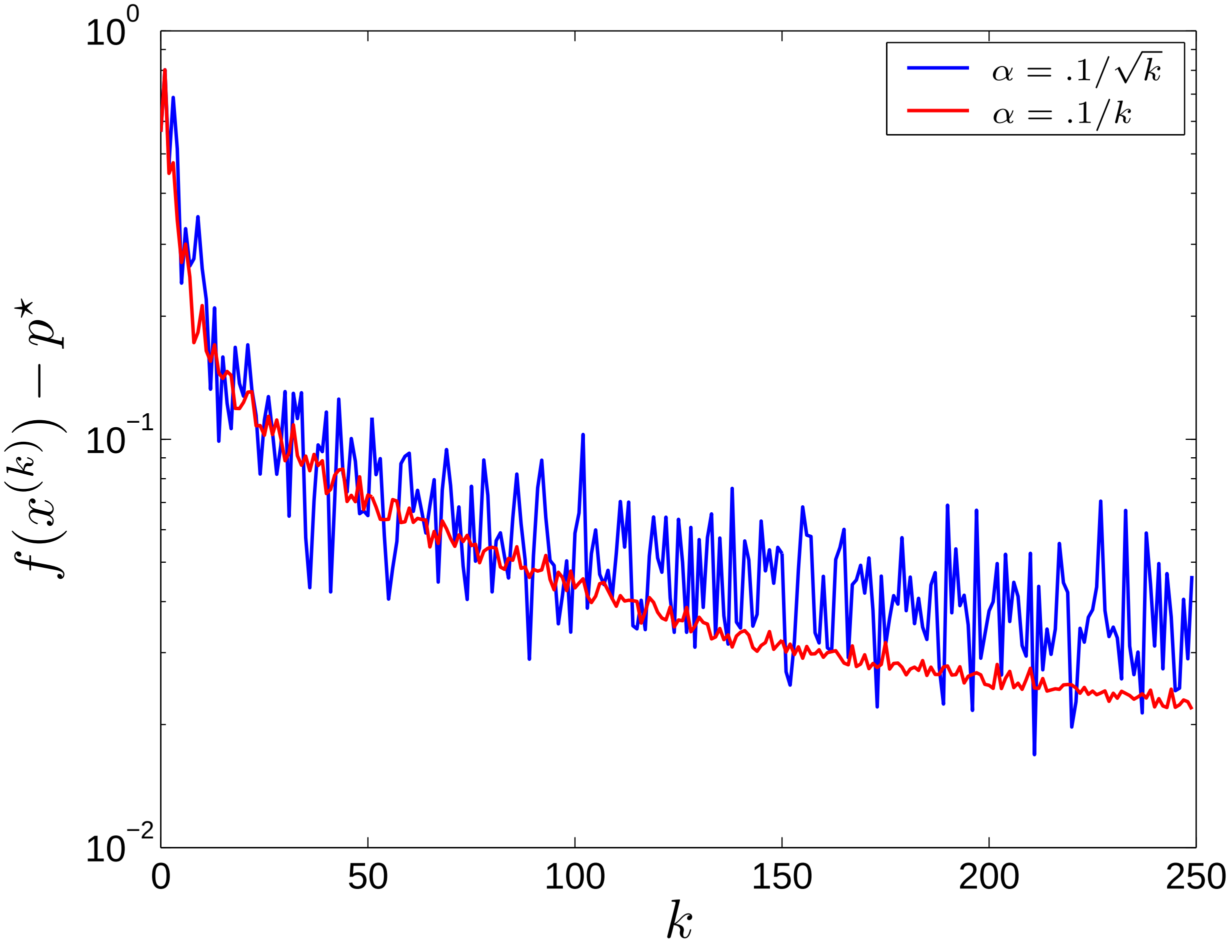
Constant step length, $h = 0.05,\ 0.02, 0.005$



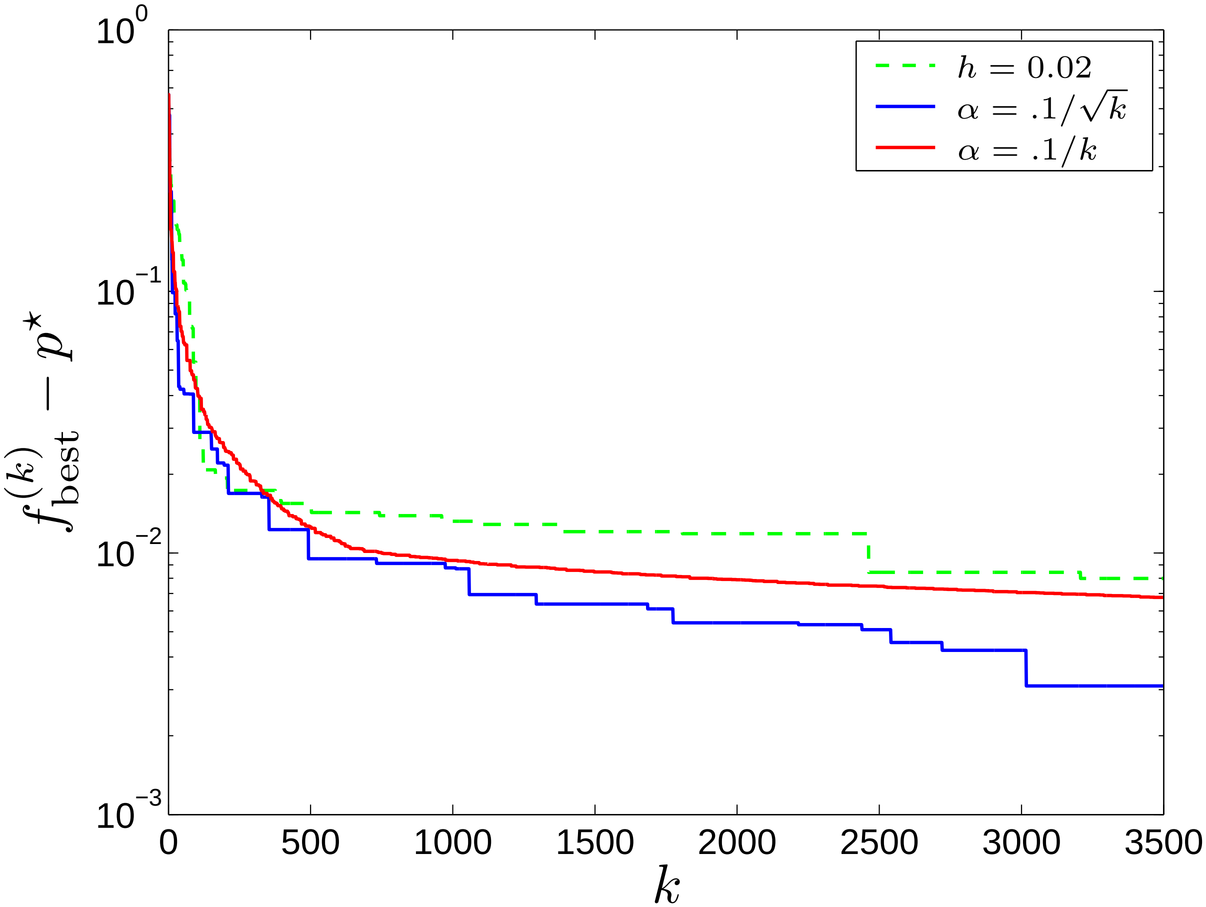
Constant step size $h = 0.05,\ 0.02,\ 0.005$



Diminishing step rule $\alpha = 0.1/\sqrt{k}$ and square summable step size rule $\alpha = 0.1/k$.



Constant step length $h = 0.02$, diminishing step size rule $\alpha = 0.1/\sqrt{k}$, and square summable step rule $\alpha = 0.1/k$



Localization methods

Localization methods

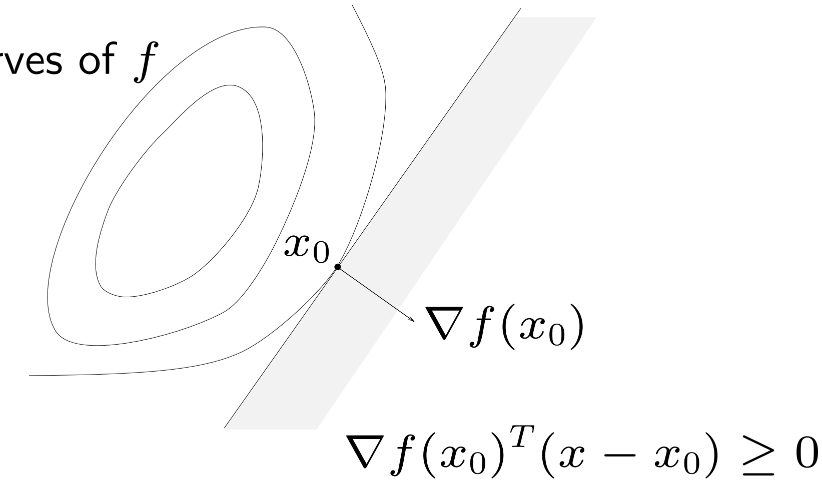
$$\text{minimize } f(x)$$

- Function $f : \mathbf{R}^n \to \mathbf{R}$ convex (and for now, differentiable)
- **oracle model:** for any x we can evaluate f and $\nabla f(x)$ (at some cost)

f **convex** means $f(x) \geq f(x_0) + \nabla f(x_0)^T (x - x_0)$ and

$$\nabla f(x_0)^T (x - x_0) \geq 0 \quad \Longrightarrow \quad f(x) \geq f(x_0)$$

$i.e.$, all points in halfspace $\nabla f(x_0)^T (x - x_0) \geq 0$ are **worse** than x_0



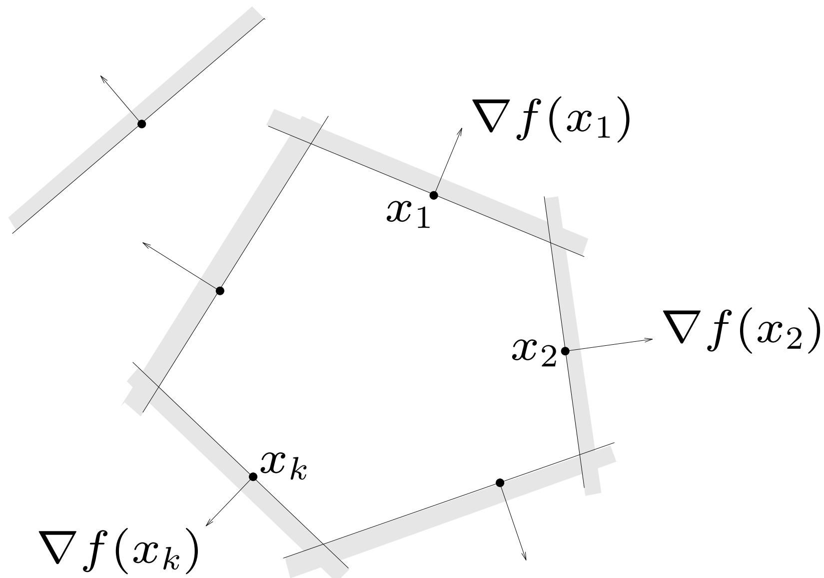
- by evaluating ∇f we rule out a halfspace in our search for $x^\star$:

$$x^\star \in \{x \mid \nabla f(x_0)^T(x - x_0) \leq 0\}$$

- **idea:** get one bit of info (on location of $x^\star$) by evaluating ∇f
- for nondifferentiable f, can replace $\nabla f(x_0)$ with any subgradient $g \in \partial f(x_0)$

Suppose we have evaluated $\nabla f(x_1), \ldots, \nabla f(x_k)$ then we know

$x^\star \in \{x \mid \nabla f(x_i)^T(x - x_i) \leq 0\}$



on the basis of $\nabla f(x_1), \ldots, \nabla f(x_k)$, we have **localized** $x^\star$ to a polyhedron

question: what is a 'good' point x_{k+1} at which to evaluate ∇f?

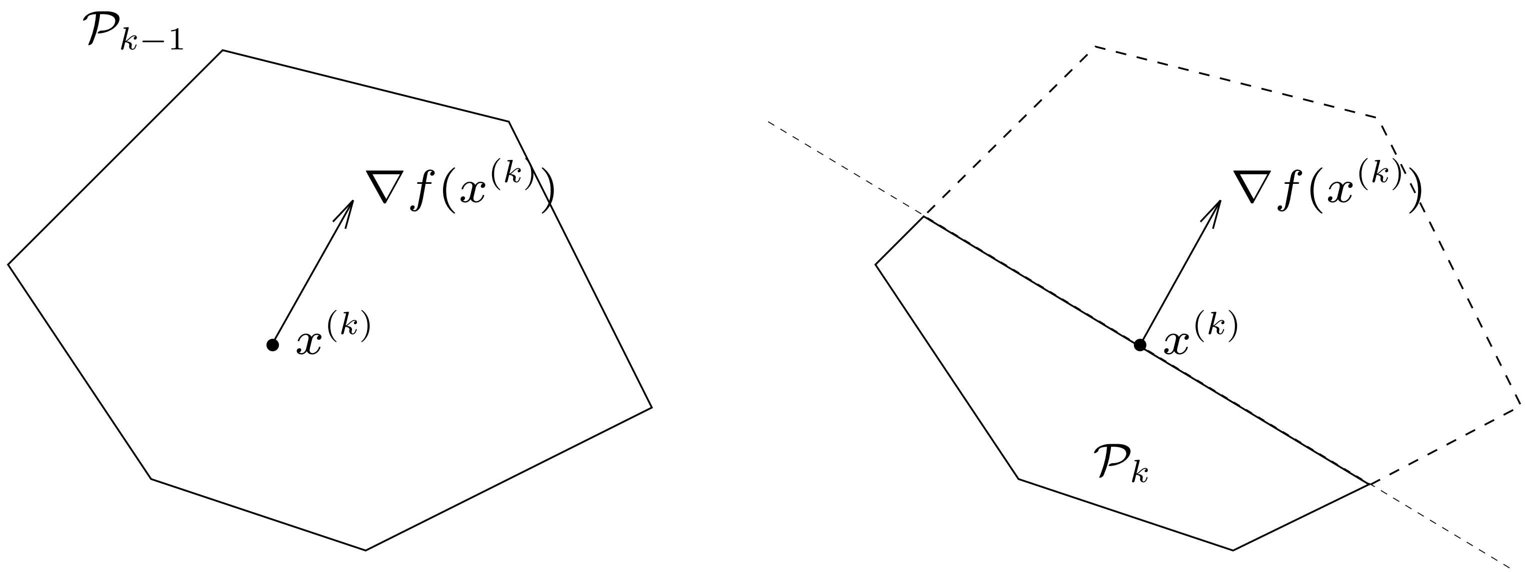
Localization algorithm

Basic **localization** (or cutting-plane) algorithm:

1. after iteration $k-1$ we know $x^\star \in \mathcal{P}_{k-1}$:

$$\mathcal{P}_{k-1} = \{x \mid \nabla f(x^{(i)})^T(x - x^{(i)}) \leq 0,\ i = 1, \ldots, k-1\}$$

2. evaluate $\nabla f(x^{(k)})$ (or $g \in \partial f(x^{(k)})$) for some $x^{(k)} \in \mathcal{P}_{k-1}$
3. $\mathcal{P}_k := \mathcal{P}_{k-1} \cap \{x \mid \nabla f(x^{(k)})^T(x - x^{(k)}) \leq 0\}$



- $\mathcal{P}_k$ gives our uncertainty of $x^\star$ at iteration k
- want to pick $x^{(k)}$ so that $\mathcal{P}_{k+1}$ is as small as possible
- clearly want $x^{(k)}$ near center of $C^{(k)}$

Example: bisection on R

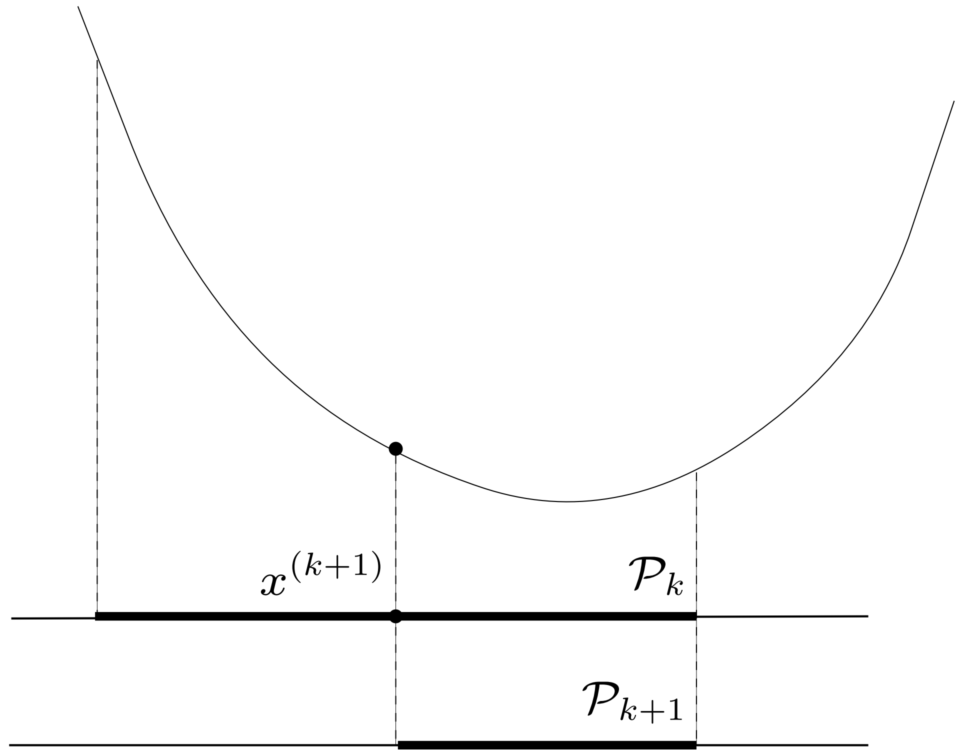
- $f : \mathbf{R} \to \mathbf{R}$
- $\mathcal{P}_k$ is interval
- obvious choice: $x^{(k+1)} :=$ midpoint$(\mathcal{P}_k)$

bisection algorithm

given interval $C = [l, u]$ containing $x^\star$

repeat

1. $x := (l + u)/2$
2. evaluate $f'(x)$
3. if $f'(x) < 0$, $l := x$; else $u := x$



$$\text{length}(\mathcal{P}_{k+1}) = u_{k+1} - l_{k+1} = \frac{u_k - l_k}{2} = (1/2)\text{length}(\mathcal{P}_k)$$

and so $\text{length}(\mathcal{P}_k) = 2^{-k}\text{length}(\mathcal{P}_0)$

interpretation:

- $\text{length}(\mathcal{P}_k)$ measures our uncertainty in $x^\star$
- uncertainty is halved at each iteration; get exactly one bit of info about $x^\star$ per iteration
- $\#$ steps required for uncertainty (in $x^\star$) $\leq \epsilon$:

$$\log_2 \frac{\text{length}(\mathcal{P}_0)}{\epsilon} = \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}$$

question:

- can bisection be extended to $\mathbf{R}^n$?
- or is it special since $\mathbf{R}$ is linear ordering?

Center of gravity algorithm

Take $x^{(k+1)} = \mathsf{CG}(\mathcal{P}_k)$ (center of gravity)

$$\mathsf{CG}(\mathcal{P}_k) = \int_{\mathcal{P}_k} x \, dx \bigg/ \int_{\mathcal{P}_k} dx$$

theorem. if $C \subseteq \mathbf{R}^n$ convex, $x_{\mathsf{cg}} = \mathsf{CG}(C)$, $g \neq 0$,

$$\mathbf{vol}\left(C \cap \{x \mid g^T(x - x_{\mathsf{cg}}) \leq 0\}\right) \leq (1 - 1/e)\, \mathbf{vol}(C) \approx 0.63\ \mathbf{vol}(C)$$

(independent of dimension n)

hence in CG algorithm, $\mathbf{vol}(\mathcal{P}_k) \leq 0.63^k\, \mathbf{vol}(\mathcal{P}_0)$

- $\mathbf{vol}(\mathcal{P}_k)^{1/n}$ measures uncertainty (in $x^\star$) at iteration k
- uncertainty reduced at least by $0.63^{1/n}$ each iteration
- from this can prove $f(x^{(k)}) \to f(x^\star)$ (later)
- max. $\#$ steps required for uncertainty $\leq \epsilon$:

$$1.51 n \log_2 \frac{\text{initial uncertainty}}{\text{final uncertainty}}$$

(cf. bisection on **R**)

advantages of CG-method

- guaranteed convergence
- number of steps proportional to dimension n, log of uncertainty reduction

disadvantages

- finding $x^{(k+1)} = \text{CG}(\mathcal{P}_k)$ is **harder** than original problem
- $\mathcal{P}_k$ becomes more complex as k increases
  (removing redundant constraints is harder than solving original problem)

(but, can modify CG-method to work)

Analytic center cutting-plane method

analytic center of polyhedron $\mathcal{P} = \{z \mid a_i^T z \preceq b_i,\ i = 1, \ldots, m\}$ is

$$\mathsf{AC}(\mathcal{P}) = \underset{z}{\operatorname{argmin}} - \sum_{i=1}^{m} \log(b_i - a_i^T z)$$

ACCPM is localization method with next query point $x^{(k+1)} = \mathsf{AC}(\mathcal{P}_k)$ (found by Newton's method)

Outer ellipsoid from analytic center

- let x^* be analytic center of $\mathcal{P} = \{z \mid a_i^T z \preceq b_i, \;\; i = 1, \dots, m\}$
- let H^* be Hessian of barrier at x^*,

$$H^* = -\nabla^2 \sum_{i=1}^m \log(b_i - a_i^T z)\Big|_{z=x^*} = \sum_{i=1}^m \frac{a_i a_i^T}{(b_i - a_i^T x^*)^2}$$

- then, $\mathcal{P} \subseteq \mathcal{E} = \{z \mid (z - x^*)^T H^* (z - x^*) \leq m^2\}$ (not hard to show)

Lower bound in ACCPM

let $\mathcal{E}^{(k)}$ be outer ellipsoid associated with $x^{(k)}$

a lower bound on optimal value $p^\star$ is

$$\begin{aligned} p^\star &\geq \inf_{z \in \mathcal{E}^{(k)}} \left(f(x^{(k)}) + g^{(k)T}(z - x^{(k)}) \right) \\ &= f(x^{(k)}) - m_k \sqrt{g^{(k)T} H^{(k)-1} g^{(k)}} \end{aligned}$$

(m_k is number of inequalities in $\mathcal{P}_k$)

gives simple stopping criterion $\sqrt{g^{(k)T} H^{(k)-1} g^{(k)}} \leq \epsilon / m_k$

Best objective and lower bound

since ACCPM isn't a descent a method, we keep track of best point found, and best lower bound

best function value so far: $u_k = \min_{i=1,\ldots,k} f(x^{(k)})$

best lower bound so far: $l_k = \max_{i=1,\ldots,k} f(x^{(k)}) - m_k \sqrt{g^{(k)T} H^{(k)-1} g^{(k)}}$

can stop when $u_k - l_k \leq \epsilon$

Basic ACCPM

given polyhedron $\mathcal{P}$ containing $x^\star$

repeat

1. compute x^*, the analytic center of $\mathcal{P}$, and H^*
2. compute $f(x^*)$ and $g \in \partial f(x^*)$
3. $u := \min\{u, f(x^*)\}$
   $l := \max\{l, f(x^*) - m\sqrt{g^T H^{*-1} g}\}$
4. add inequality $g^T(z - x^*) \leq 0$ to $\mathcal{P}$

until $u - l < \epsilon$

here m is number of inequalities in $\mathcal{P}$

Dropping constraints

add an inequality to $\mathcal{P}$ each iteration, so centering gets harder, more storage as algorithm progresses

schemes for dropping constraints from $\mathcal{P}^{(k)}$:

- remove all redundant constraints (expensive)
- remove some constraints known to be redundant
- remove constraints based on some relevance ranking

Dropping constraints in ACCPM

x^* is AC of $\mathcal{P} = \{x \mid a_i^T x \leq b_i,\ i = 1, \ldots, m\}$, H^* is barrier Hessian at x^*

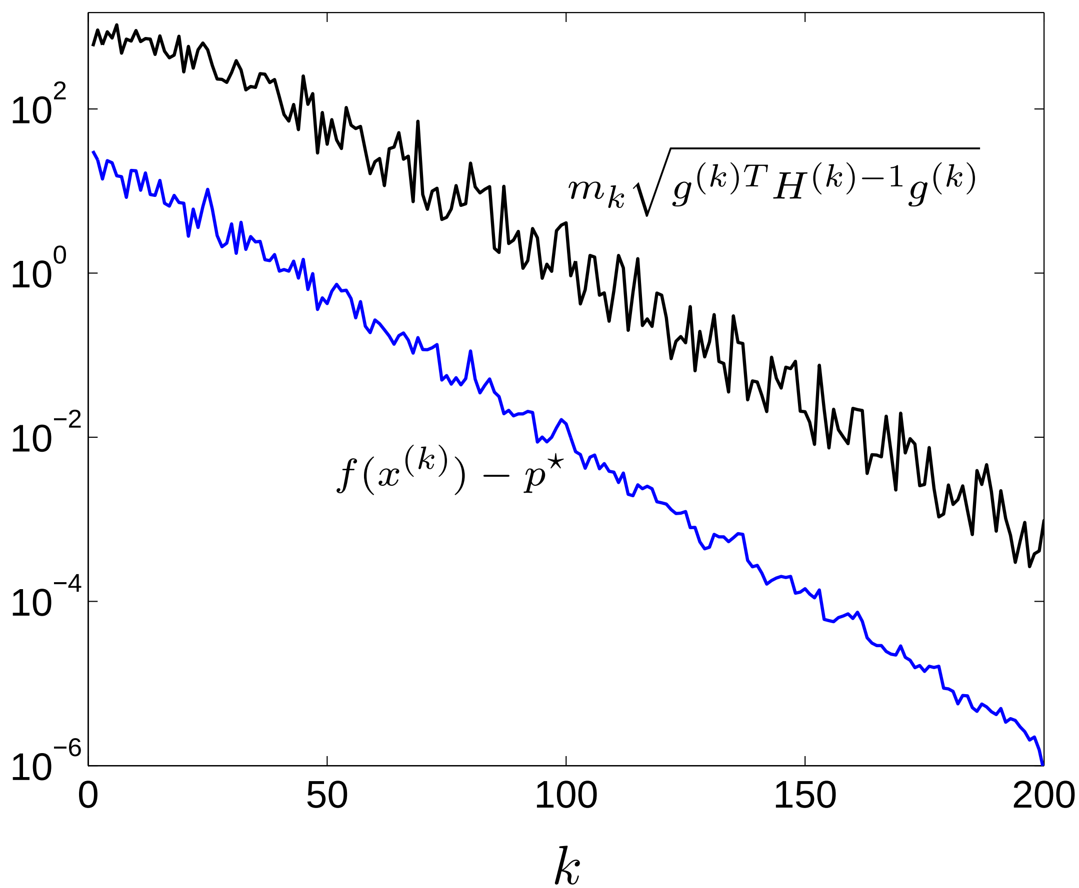
define (ir)relevance measure $\eta_i = \dfrac{b_i - a_i^T x^*}{\sqrt{a_i^T H^{*-1} a_i}}$

- η_i / m is normalized distance from hyperplane $a_i^T x = b_i$ to outer ellipsoid
- if $\eta_i \geq m$, then constraint $a_i^T x \leq b_i$ is redundant

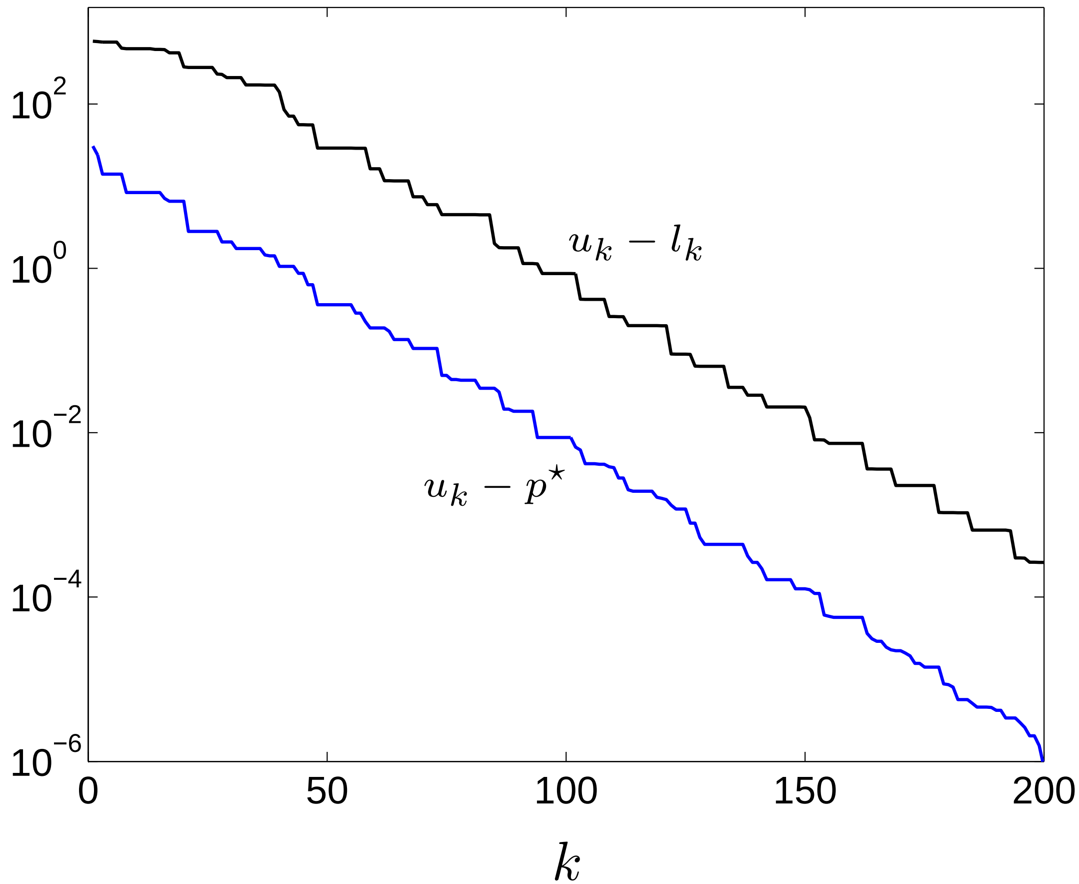
Example

PWL objective, $n = 10$ variables, $m = 100$ terms

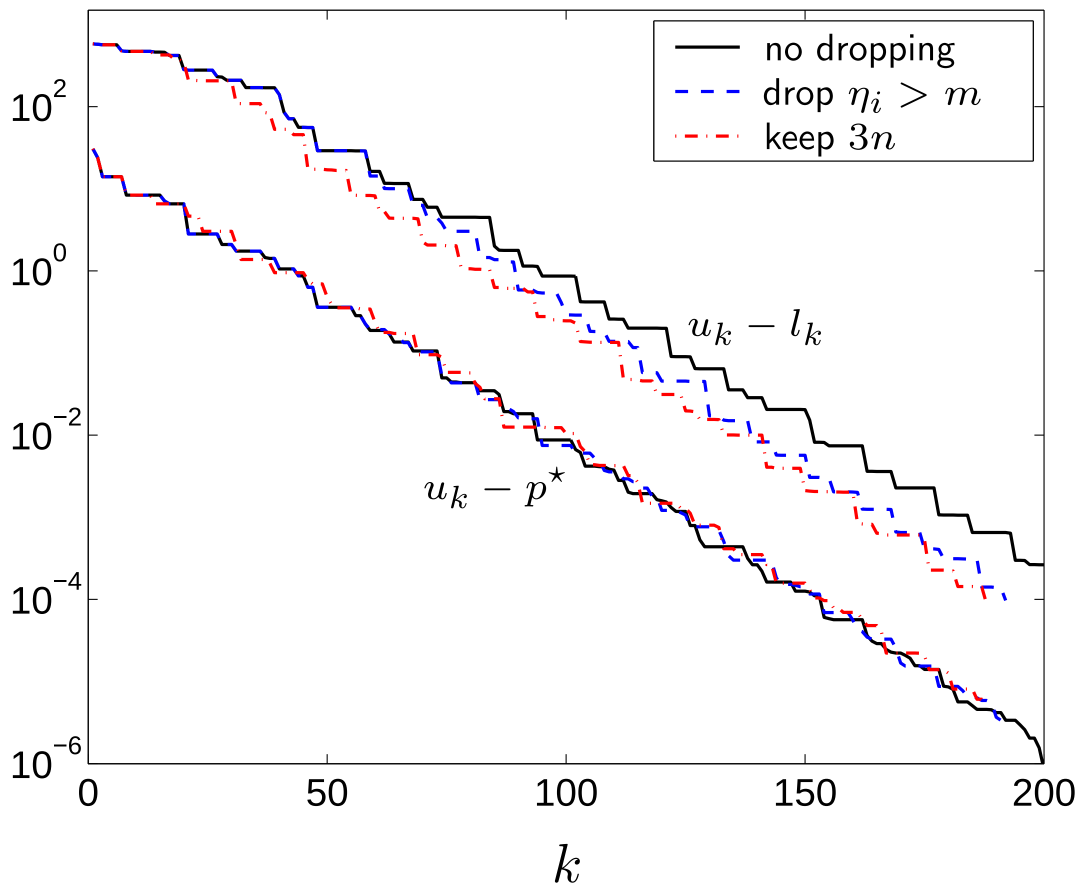
simple ACCPM: $f(x^{(k)})$ and lower bound $f(x^{(k)}) - m\sqrt{g^{(k)T}H^{(k)-1}g^{(k)}}$



simple ACCPM: u_k (best objective value) and l_k (best lower bound)



ACCPM with constraint dropping



. . . constraint dropping actually **improves** convergence (!)

ACCPM with constraint dropping

number of inequalities in $\mathcal{P}$: