Convex Optimization Theory and Applications

Topic 18 - Successive Approximation Based Optimization

Li Li

Department of Automation
Tsinghua University

Fall, 2009-2021.

18.0. Outline

18.1. Successive Linear Programming

repetitively linearize objective functions and constraints

implementations usually invlove trust regions which prevent any iteration from straying farther than the linear appproximations would be valid

some implementations use penalty functions to maintain feasibility with respect to the true constratins

18.1. Successive Linear Programming

Advantages of SLP:

- relatively fast
- theoretically simple
- commercial packages are available

Disadvantanges of SLP:

- method frequently fails in practice due to the highly nonlinear nature of the problem
- poor choice for trust region may allow infeasible LPs to be genreated

18.1. Successive Linear Programming

例：
$$\begin{cases} \min f(x) = 4x_1^2 + (x_2 - 2)^2 \\ s.t. \quad -2 \le x_1 \le 2 \\ \qquad -1 \le x_2 \le 1 \end{cases}$$

解：取初始点$x^{(1)} = (-2, -1)^T$

第一次迭代 $\nabla f(x^{(1)}) = (-16, -6)^T$

求
$$\begin{cases} \min \nabla f(x^{(1)})^T x = -16x_1 - 6x_2 \\ s.t. \quad -2 \le x_1 \le 2 \\ \qquad -1 \le x_2 \le 1 \end{cases}$$

得$y^{(1)} = (2, 1)^T$.

18.1. Successive Linear Programming

从$x^{(1)}$出发，沿$y^{(1)}-x^{(1)}$作一维搜索

$$\begin{cases}\min f(x^{(1)}+\lambda(y^{(1)}-x^{(1)})) \\ s.t. \quad 0\le\lambda\le 1\end{cases}$$

得$\lambda_1=0.56$

$$\therefore x^{(2)}=x^{(1)}+\lambda_1(y^{(1)}-x^{(1)})=(0.24,0.12)^T$$

第二次迭 代

求
$$\begin{cases}\min \nabla f(x^{(2)})^T x=1.92x_1-3.76x_2 \\ s.t. \quad -2\le x_1\le 2 \\ \qquad -1\le x_2\le 1\end{cases}$$

得$y^{(2)}=(-2,1)^T$.

18.1. Successive Linear Programming

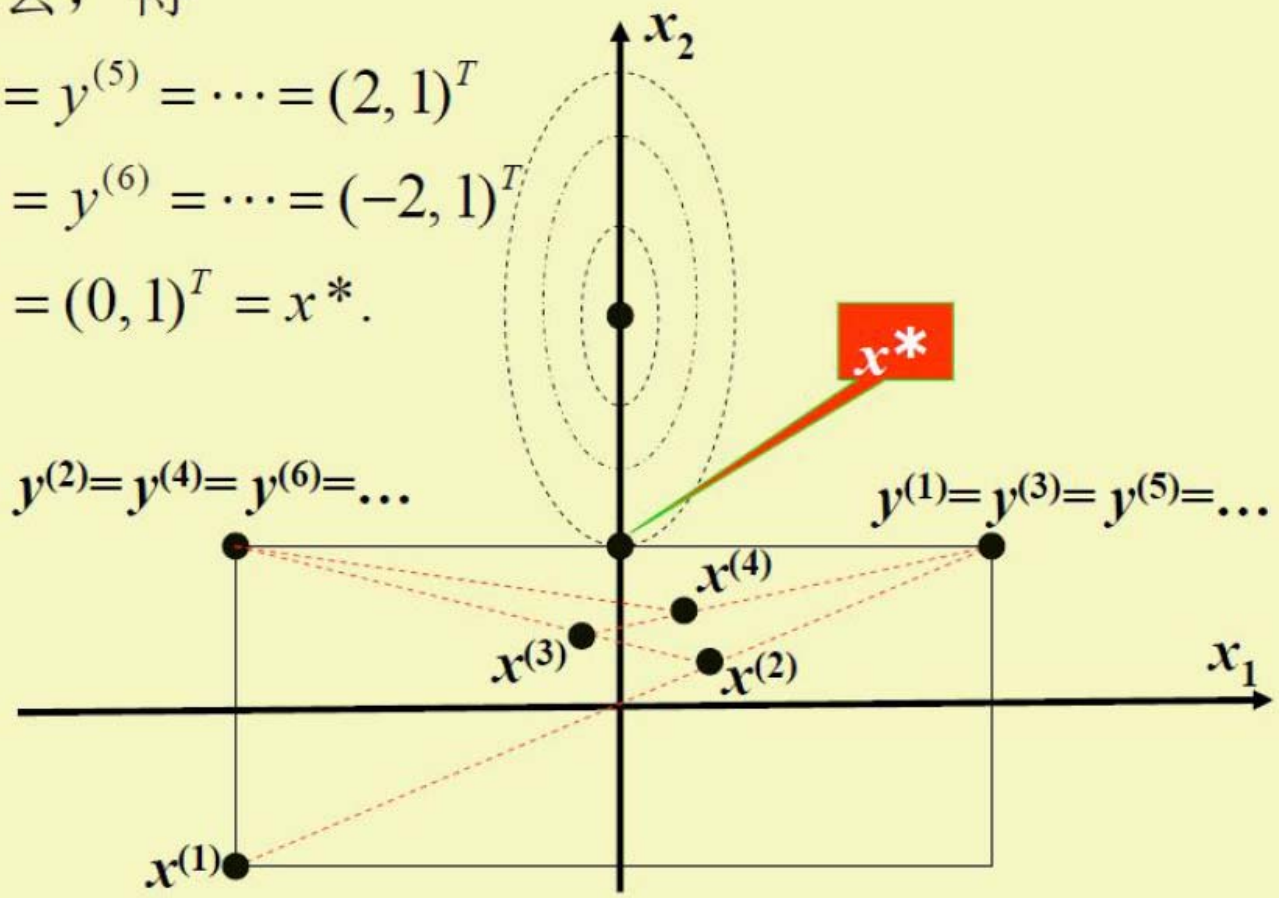
从$x^{(2)}$出发，沿$y^{(2)}-x^{(2)}$作一维搜索，得$x^{(3)}$，

这样继续下去，得

$$y^{(1)}=y^{(3)}=y^{(5)}=\cdots=(2,1)^T$$

$$y^{(2)}=y^{(4)}=y^{(6)}=\cdots=(-2,1)^T$$

显然$\lim_{k\to\infty} x^{(k)}=(0,1)^T=x^*$.



$$\text{Maximize:} \quad 2x + y$$

$$\text{Subject to:} \quad x^2 + y^2 \leq 25$$

$$x^2 - y^2 \leq 7$$

and

$$x \geq 0$$

$$y \geq 0$$

with an initial starting point of $(x_c, y_c) = (2, 2)$. Figure 8.7 shows the two nonlinear constraints and one objective function contour with an objective value of 10.

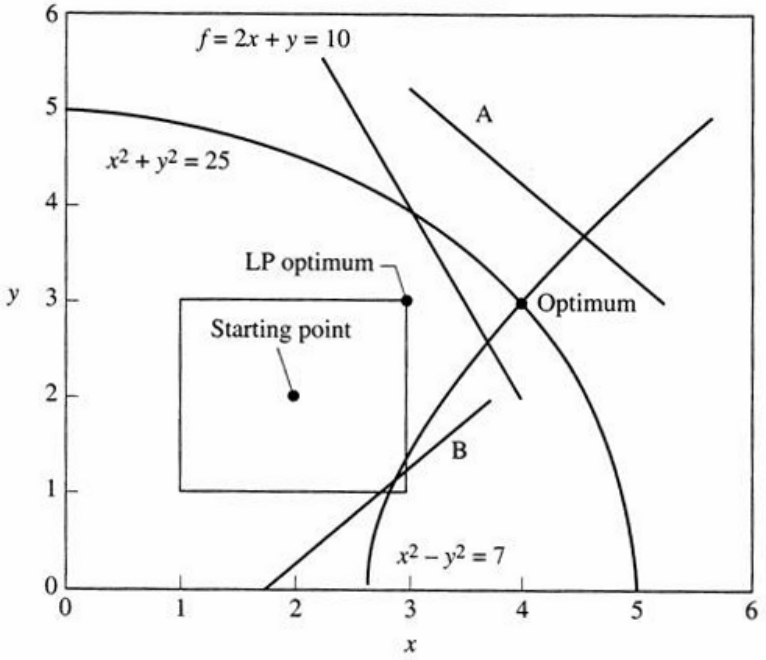


FIGURE 8.7
SLP example with linear objective, nonlinear constraints. Line A is the linearization of $x^2 + y^2 \leq 25$ and line B is the linearization of $x^2 - y^2 \leq 7$.

Use $x_c = 2$ $y_c = 2$ for linearization

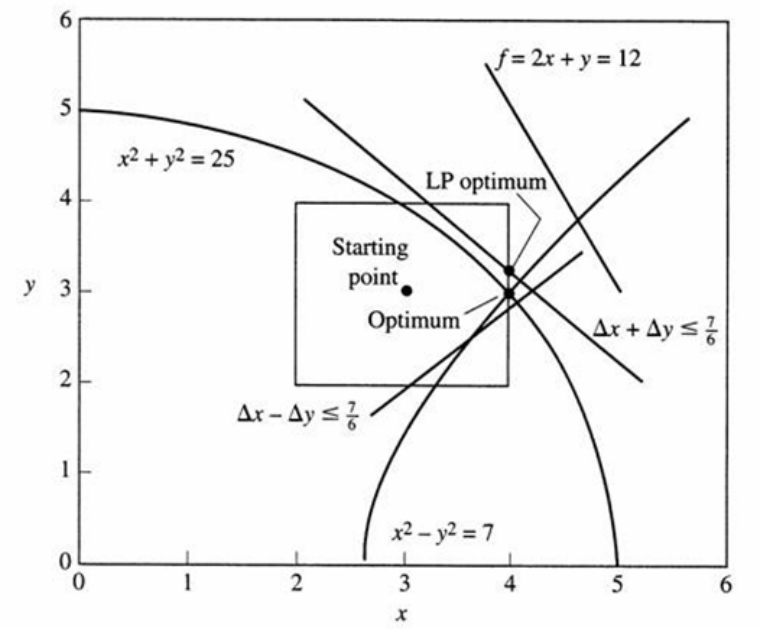
Maximize: $2x_c + y_c + 2\Delta x + \Delta y = 2\Delta x + \Delta y + 6$

Subject to: $x_c^2 + y_c^2 + 2x_c\Delta x + 2y_c\Delta y = 4\Delta x + 4\Delta y + 8 \le 25$

$x_c^2 - y_c^2 + 2x_c\Delta x - 2y_c\Delta y = 4\Delta x - 4\Delta y \le 7$

$2 + \Delta x \ge 0, \quad 2 + \Delta y \ge 0$

$-1 \le \Delta x \le 1, \quad -1 \le \Delta y \le 1$ (step bounds)



Maximize: $2\Delta x + \Delta y$

Subject to: $\Delta x + \Delta y \le 4.25$

$\Delta x - \Delta y \le 1.75$

and

$-1 \le \Delta x \le 1$

$-1 \le \Delta y \le 1$

Maximize: $2\Delta x + \Delta y$

Subject to: $\Delta x + \Delta y \le \frac{7}{6}$

$\Delta x - \Delta y \le \frac{7}{6}$

and

$-1 \le \Delta x \le 1$

$-1 \le \Delta y \le 1$

The feasible region can be seen in Figure 8.8 and the optimal solution is at $(\Delta x, \Delta y) = (1, \frac{1}{6})$ or $(x_n, y_n) = (4, 3.167)$. This point is at the intersection of the constraints $\Delta x + \Delta y \le \frac{7}{6}$ and $\Delta x = 1$, so one step bound is still active at the LP optimum.

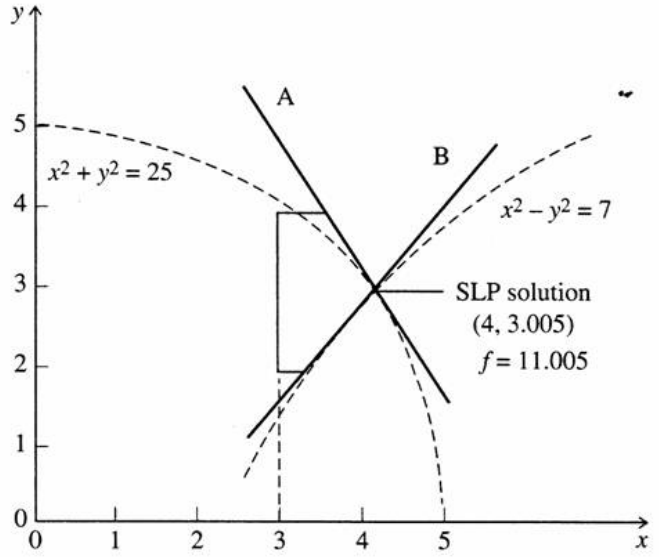


FIGURE 8.9
The optimal point after solving the third SLP subproblem. A is the linearization of $x^2 + y^2 = 25$ and B is the linearization of $x^2 - y^2 = 7$.

18.2. Frank-Wolfe Method

The Frank-Wolfe method, also called conditional gradient method, uses a local linear expansion of f:

$$s^{(k-1)} \in \operatorname*{argmin}_{s \in C} \nabla f(x^{(k-1)})^T s$$

$$x^{(k)} = (1 - \gamma_k) x^{(k-1)} + \gamma_k s^{(k-1)}$$

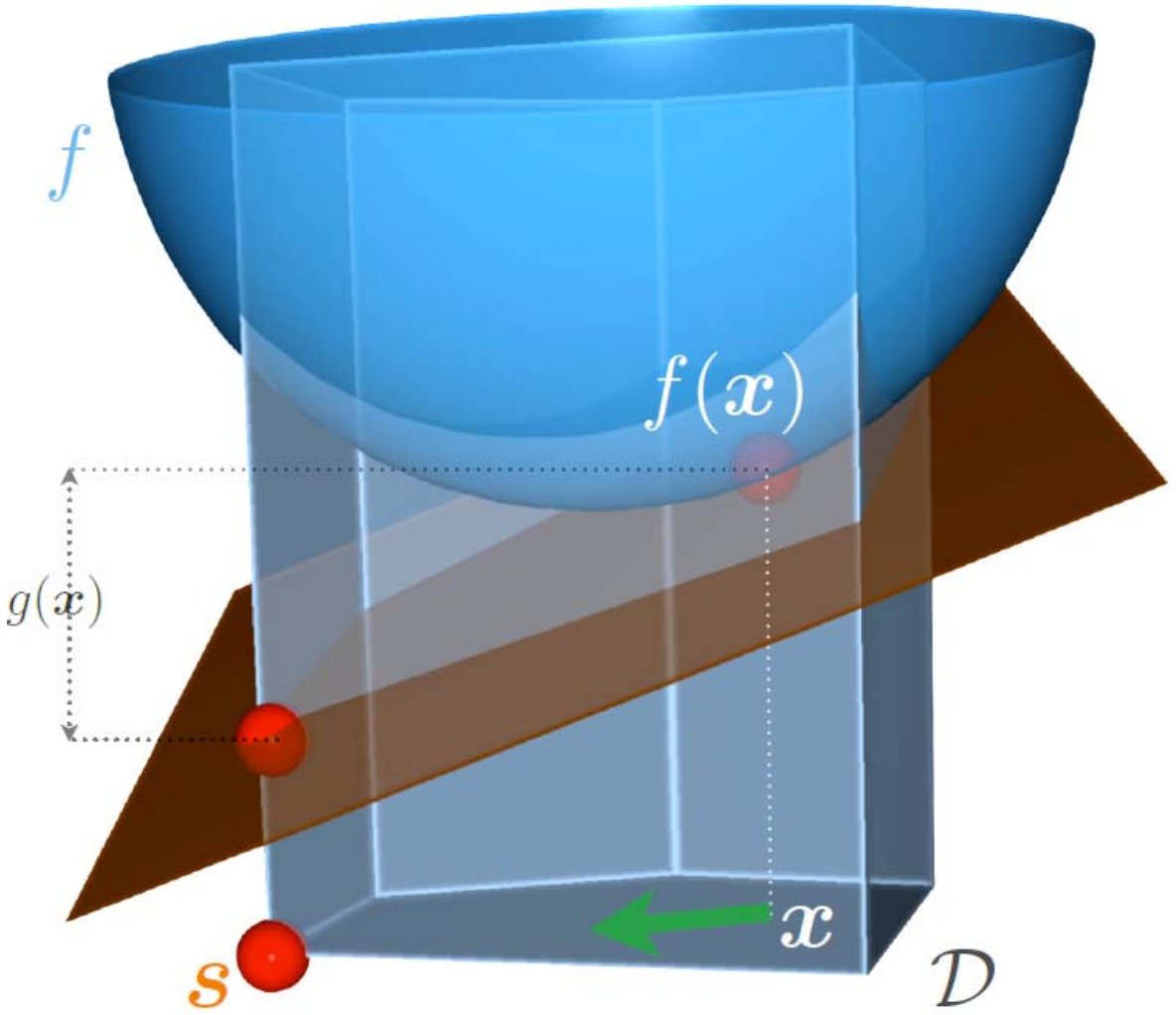
Note that there is no projection; update is solved directly over C

Default step sizes: $\gamma_k = 2/(k+1)$, $k = 1, 2, 3, \ldots$. Note for any $0 \leq \gamma_k \leq 1$, we have $x^{(k)} \in C$ by convexity. Can rewrite update as

$$x^{(k)} = x^{(k-1)} + \gamma_k (s^{(k-1)} - x^{(k-1)})$$

i.e., we are moving less and less in the direction of the linearization minimizer as the algorithm proceeds

18.2. Frank-Wolfe Method



M. Jaggi (2011), Revisiting Frank-Wolfe: projection-free sparse convex optimization

18.2. Frank-Wolfe Method

Frank-Wolfe iterations admit a very natural duality gap:

$$g(x^{(k)}) = \nabla f(x^{(k)})^T (x^{(k)} - s^{(k)})$$

Claim: it holds that $f(x^{(k)}) - f^\star \leq g(x^{(k)})$

Proof: by the first-order condition for convexity

$$f(s) \geq f(x^{(k)}) + \nabla f(x^{(k)})^T (s - x^{(k)})$$

Minimizing both sides over all $s \in C$ yields

$$\begin{aligned} f^\star &\geq f(x^{(k)}) + \min_{s \in C} \nabla f(x^{(k)})^T (s - x^{(k)}) \\ &= f(x^{(k)}) + \nabla f(x^{(k)})^T (s^{(k)} - x^{(k)}) \end{aligned}$$

Rearranged, this gives the duality gap above

18.2. Frank-Wolfe Method

Why do we call it "duality gap"? Rewrite original problem as

$$\min_x \ f(x) + I_C(x)$$

where I_C is the indicator function of C. The dual problem is

$$\max_u \ -f^*(u) - I_C^*(-u)$$

where I_C^* is the support function of C. Duality gap at x, u is

$$f(x) + f^*(u) + I_C^*(-u) \geq x^T u + I_C^*(-u)$$

Evaluated at $x = x^{(k)}$, $u = \nabla f(x^{(k)})$, this gives

$$\nabla f(x^{(k)})^T x^{(k)} + \max_{s \in C} \ -\nabla f(x^{(k)})^T s = \nabla f(x^{(k)})^T (x^{(k)} - s^{(k)})$$

which is our gap

18.2. Frank-Wolfe Method

Following Jaggi (2011), define the curvature constant of f over C:

$$M = \max_{\substack{\gamma \in [0,1] \\ x,s,y \in C \\ y=(1-\gamma)x+\gamma s}} \frac{2}{\gamma^2}\Big(f(y) - f(x) - \nabla f(x)^T(y-x)\Big)$$

Note that $M = 0$ for linear f, and $f(y) - f(x) - \nabla f(x)^T(y-x)$ is called the Bregman divergence, defined by f

Theorem: The Frank-Wolfe method using standard step sizes $\gamma_k = 2/(k+1)$, $k = 1, 2, 3, \ldots$ satisfies

$$f(x^{(k)}) - f^\star \leq \frac{2M}{k+2}$$

Thus number of iterations needed for $f(x^{(k)}) - f^\star \leq \epsilon$ is $O(1/\epsilon)$

This matches the sublinear rate for projected gradient descent for Lipschitz ∇f, but how do the assumptions compare?

For Lipschitz ∇f with constant L, recall

$$f(y) - f(x) - \nabla f(x)^T (y - x) \leq \frac{L}{2} \|y - x\|_2^2$$

Maximizing over all $y = (1 - \gamma)x + \gamma s$, and multiplying by $2/\gamma^2$,

$$\begin{aligned} M \leq & \max_{\substack{\gamma \in [0,1] \\ x,s,y \in C \\ y = (1-\gamma)x + \gamma s}} \frac{2}{\gamma^2} \cdot \frac{L}{2} \|y - x\|_2^2 \\ = & \max_{x,s \in C} L \|x - s\|_2^2 = L \cdot \mathrm{diam}^2(C) \end{aligned}$$

Hence assuming a bounded curvature is basically no stronger than what we assumed for projected gradient

18.2. Frank-Wolfe Method

The key inequality used to prove the Frank-Wolfe convergence rate:

$$f(x^{(k)}) \leq f(x^{(k-1)}) - \gamma_k g(x^{(k-1)}) + \frac{\gamma_k^2}{2} M$$

Here $g(x) = \max_{s \in C} \nabla f(x)^T (x - s)$ is duality gap defined earlier

Proof: write $x^+ = x^{(k)}$, $x = x^{(k-1)}$, $s = s^{(k-1)}$, $\gamma = \gamma_k$. Then

$$\begin{aligned} f(x^+) &= f\big(x + \gamma(s - x)\big) \\ &\leq f(x) + \gamma \nabla f(x)^T (s - x) + \frac{\gamma^2}{2} M \\ &= f(x) - \gamma g(x) + \frac{\gamma^2}{2} M \end{aligned}$$

Second line used definition of M, and third line the definition of g

18.2. Frank-Wolfe Method

The proof of the convergence result is now straightforward. Denote by $h(x) = f(x) - f^\star$ the suboptimality gap at x. Basic inequality:

$$\begin{aligned} h(x^{(k)}) &\leq h(x^{(k-1)}) - \gamma_k g(x^{(k-1)}) + \frac{\gamma_k^2}{2} M \\ &\leq h(x^{(k-1)}) - \gamma_k h(x^{(k-1)}) + \frac{\gamma_k^2}{2} M \\ &= (1 - \gamma_k) h(x^{(k-1)}) + \frac{\gamma_k^2}{2} M \end{aligned}$$

where in the second line we used $g(x^{(k-1)}) \geq h(x^{(k-1)})$

To get the desired result we use induction:

$$h(x^{(k)}) \leq \left(1 - \frac{2}{k+1}\right) \frac{2M}{k+1} + \left(\frac{2}{k+1}\right)^2 \frac{M}{2} \leq \frac{2M}{k+2}$$

18.2. Frank-Wolfe Method

Frank-Wolfe updates are affine invariant: for nonsingular matrix A, define $x = Ax'$, $F(x') = f(Ax')$, consider Frank-Wolfe on F:

$$s' = \operatorname*{argmin}_{z \in A^{-1}C} \nabla F(x')^T z$$
$$(x')^+ = (1-\gamma)x' + \gamma s'$$

Multiplying by A produces same Frank-Wolfe update as that from f. Convergence analysis is also affine invariant: curvature constant

$$M = \max_{\substack{\gamma \in [0,1] \\ x', s', y' \in A^{-1}C \\ y' = (1-\gamma)x' + \gamma s'}} \frac{2}{\gamma^2}\Big(F(y') - F(x') - \nabla F(x')^T(y' - x')\Big)$$

matches that of f, because $\nabla F(x')^T(y' - x') = \nabla f(x)^T(y - x)$

18.2. Frank-Wolfe Method

Jaggi (2011) also analyzes inexact Frank-Wolfe updates: suppose we choose $s^{(k-1)}$ so that

$$\nabla f(x^{(k-1)})^T s^{(k-1)} \leq \min_{s \in C} \nabla f(x^{(k-1)})^T s + \frac{M\gamma_k}{2} \cdot \delta$$

where $\delta \geq 0$ is our inaccuracy parameter. Then we basically attain the same rate

Theorem: Frank-Wolfe using step sizes $\gamma_k = 2/(k+1)$, $k = 1, 2, 3, \ldots$, and inaccuracy parameter $\delta \geq 0$, satisfies

$$f(x^{(k)}) - f^\star \leq \frac{2M}{k+1}(1+\delta)$$

Note: the optimization error at step k is $M\gamma_k/2 \cdot \delta$. Since $\gamma_k \to 0$, we require the errors to vanish

18.2. Frank-Wolfe Method

Two important variants of Frank-Wolfe:

- Line search: instead of using standard step sizes, use

$$\gamma_k = \underset{\gamma \in [0,1]}{\operatorname{argmin}} \; f\big(x^{(k-1)} + \gamma(s^{(k-1)} - x^{(k-1)})\big)$$

  at each $k = 1, 2, 3, \ldots$. Or, we could use backtracking

- Fully corrective: directly update according to

$$x^{(k)} = \underset{y}{\operatorname{argmin}} \; f(y) \;\; \text{subject to} \;\; y \in \operatorname{conv}\{x^{(0)}, s^{(0)}, \ldots, s^{(k-1)}\}$$

Both variants lead to the same $O(1/\epsilon)$ iteration complexity

Another popular variant: away steps, which get linear convergence under strong convexity

18.2. Frank-Wolfe Method

What happens when $C = \{x : \|x\| \leq t\}$ for a norm $\|\cdot\|$? Then

$$\begin{aligned} s &\in \underset{\|s\| \leq t}{\operatorname{argmin}} \ \nabla f(x^{(k-1)})^T s \\ &= -t \cdot \left(\underset{\|s\| \leq 1}{\operatorname{argmax}} \ \nabla f(x^{(k-1)})^T s \right) \\ &= -t \cdot \partial \|\nabla f(x^{(k-1)})\|_* \end{aligned}$$

where $\|\cdot\|_*$ denotes the corresponding dual norm. That is, if we know how to compute subgradients of the dual norm, then we can easily perform Frank-Wolfe steps

A key to Frank-Wolfe: this can often be simpler or cheaper than projection onto $C = \{x : \|x\| \leq t\}$

18.2. Frank-Wolfe Method

For the ℓ_1-regularized problem

$$\min_x \ f(x) \ \text{ subject to } \ \|x\|_1 \leq t$$

we have $s^{(k-1)} \in -t\partial\|\nabla f(x^{(k-1)})\|_\infty$. Frank-Wolfe update is thus

$$i_{k-1} \in \operatorname*{argmax}_{i=1,\ldots,p} \left|\nabla_i f(x^{(k-1)})\right|$$
$$x^{(k)} = (1-\gamma_k)x^{(k-1)} - \gamma_k t \cdot \operatorname{sign}\big(\nabla_{i_{k-1}} f(x^{(k-1)})\big) \cdot e_{i_{k-1}}$$

Like greedy coordinate descent! (But with diminshing steps)

Note: this is a lot simpler than projection onto the ℓ_1 ball, though both require $O(n)$ operations

18.2. Frank-Wolfe Method

For the ℓ_p-regularized problem

$$\min_x \ f(x) \ \text{ subject to } \ \|x\|_p \leq t$$

for $1 \leq p \leq \infty$, we have $s^{(k-1)} \in -t\partial\|\nabla f(x^{(k-1)})\|_q$, where p, q are dual, i.e., $1/p + 1/q = 1$. Claim: can choose

$$s_i^{(k-1)} = -\alpha \cdot \text{sign}\big(\nabla f_i(x^{(k-1)})\big) \cdot \big|\nabla f_i(x^{(k-1)})\big|^{p/q}, \quad i = 1, \ldots, n$$

where α is a constant such that $\|s^{(k-1)}\|_q = t$ (check this!), and then Frank-Wolfe updates are as usual

Note: this is a lot simpler projection onto the ℓ_p ball, for general p! Aside from special cases ($p = 1, 2, \infty$), these projections cannot be directly computed (must be treated as an optimization)

18.2. Frank-Wolfe Method

For the trace-regularized problem

$$\min_X \ f(X) \ \text{ subject to } \ \|X\|_{\text{tr}} \leq t$$

we have $S^{(k-1)} \in -t\partial\|\nabla f(X^{(k-1)})\|_{\text{op}}$. Claim: can choose

$$S^{(k-1)} = -t \cdot uv^T$$

where u, v are leading left and right singular vectors of $\nabla f(X^{(k-1)})$ (check this!), and then Frank-Wolfe updates are as usual

Note: this substantially simpler and cheaper than projection onto the trace norm ball, which requires a singular value decomposition!

18.2. Frank-Wolfe Method

- ℓ_1 norm: Frank-Wolfe update scans for maximum of gradient; proximal operator soft-thresholds the gradient step; both use $O(n)$ flops
- ℓ_p norm: Frank-Wolfe update computes raises each entry of gradient to power and sums, in $O(n)$ flops; proximal operator not generally directly computable
- Trace norm: Frank-Wolfe update computes top left and right singular vectors of gradient; proximal operator soft-thresholds the gradient step, requiring a singular value decomposition

Various other constraints yield efficient Frank-Wolfe updates, e.g., special polyhedra or cone constraints, sum-of-norms (group-based) regularization, atomic norms. See Jaggi (2011)

18.3. Sequential Quadratic Programming

SQP

- Considered by some to be the best general nonlinear programming algorithm
- Repetitively approximates nonlinear objective function with quadratic function and nonlinear constraints with linear constraints
- Uses line search rather than QP step for each iteration
- *Inequality constrained Quadratic Programming* (IQP) keeps all inequality constraints
- *Equality constrained Quadratic Programming* (EQP) only keeps equality constraints by utilizing and active set strategy
- SQP is an *Infeasible Path* method

18.3. Sequential Quadratic Programming

Advantages of QP:

- handles quadratic objective functions which are popular in practice
- handles general nonlinear programming problems
- faster than many existing algorithms
- commercial codes are available

Disadvanges of quadratic programming:

- many problems have nonlinear constraints, linearization of constraints may produce an infeasible QP
- cannot handle problems with non-psd matrix problem
- requires a lot of meomry for large problems to store the approximation Hessian matrix

18.3. Sequential Quadratic Programming

数值计算的时候还是用线性等式约束的 Newton 法

$$\min_x f(x)$$
$$\text{subject to } c(x) = 0$$

has Lagrangian $\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x)$ and f.o. KKT gives

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) - A^T(x)\lambda = 0$$
$$\nabla_\lambda \mathcal{L}(x, \lambda) = c(x) = 0$$

where $A^T(x) = [\nabla c_1(x), \nabla c_2(x), \ldots, \nabla c_m(x)]$

What is the difference from our previous study?

18.3. Sequential Quadratic Programming

$$\min_x f(x)$$
$$\text{subject to } c(x) = 0$$

has Lagrangian $\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x)$ and f.o. KKT gives

$$F(x, \lambda) = \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ c(x) \end{bmatrix} = \begin{bmatrix} \nabla f(x) - A^T(x)\lambda \\ c(x) \end{bmatrix} = 0$$

where $A^T(x) = [\nabla c_1(x), \nabla c_2(x), \ldots, \nabla c_m(x)]$

Nonlinear equations problem (Chapter 11), $F(x, \lambda) = 0$.
Main idea: use Newton's method to solve the KKT system.
(Newton method p_{k+1}: $\arg_p F(x_k) + F'(x_k)p = 0$)

18.3. Sequential Quadratic Programming

$F(x_k) + F'(x_k)p = 0$ gives the Newton-KKT system

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k & -A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} p_k \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f_k + A_k^T \lambda_k \\ -c_k \end{bmatrix}$$

Assumptions for a well defined solution of the Newton-KKT system at $(x, \lambda) = (x_k, \lambda_k)$

1. The Jacobian $A(x)$ has full rank. (LICQ)
2. $d^T \nabla^2_{xx}\mathcal{L}(x, \lambda)d > 0$ for $A(x)d = 0$. (holds close to solution if s.o. suf KKT condition is satisfied at solution)

If s.o. suf. condition satisfied, solution of KKT-system is a solution of the original one. Basic idea for EQP (if active set is known)

18.3. Sequential Quadratic Programming

We can provide an alternative view in terms of linear equality constained Newton method

Replace the problem

$$\min_x f(x)$$
$$\text{subject to } c(x) = 0$$

with

$$\min_{x,\lambda} \mathcal{L}(x, \lambda)$$
$$\text{subject to } c(x) = 0$$

with $p = x - x_k$ we make the approximations around x_k, λ_k

$$\mathcal{L}(x, \lambda) \approx \mathcal{L}(x_k, \lambda_k) + \nabla_x \mathcal{L}^T(x_k, \lambda_k)p + \frac{1}{2} p^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)p$$
$$c(x) \approx A(x_k)p + c(x_k)$$

18.3. Sequential Quadratic Programming

Giving the problem

$$\min_p \; f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) p$$
$$\text{subject to } A(x_k)p + c(x_k) = 0$$

Note that $\mathcal{L}(x_k, \lambda_k) + \nabla_x \mathcal{L}^T(x_k, \lambda_k)p = f(x_k) + \nabla f(x_k)^T p$

if the assumptions hold, has a unique solution satisfying (f.o. KKT)

$$\nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)p + \nabla f(x_k) - A^T(x_k)\bar{\lambda} = 0$$
$$A(x_k)p + c(x_k) = 0$$

$$\begin{bmatrix} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) & -A^T(x_k) \\ A(x_k) & 0 \end{bmatrix} \begin{bmatrix} p \\ \bar{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ -c(x_k) \end{bmatrix}$$

18.3. Sequential Quadratic Programming

Adding $A^T(x_k)\lambda_k$ on both sides of first row and writing $\bar{\lambda} - \lambda_k = p_\lambda$

$$\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)p - A^T(x_k)(\bar{\lambda} + \lambda_k) = -\nabla f(x_k) + A^T(x_k)\lambda_k$$

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k) & -A^T(x_k) \\ A(x_k) & 0 \end{bmatrix} \begin{bmatrix} p \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) + A^T(x_k)\lambda_k \\ -c(x_k) \end{bmatrix}$$

with $\bar{\lambda} = \lambda$ (nonsigularity of problem), same solution p!

Such approach is also called Lagrange-Newton method.

18.3. Sequential Quadratic Programming

We can also get a more general SQP case

The problem

$$\begin{aligned} \min_x \ & f(x) \\ \text{subject to } & c_i(x) = 0,\ i \in \mathcal{E} \\ & c_i(x) \geq 0,\ i \in \mathcal{I} \end{aligned}$$

is linearized at each iteration giving the QP subproblems

$$\begin{aligned} \min_p \ & f(x_k) + \nabla f^T(x_k)p + \frac{1}{2}p^T\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)p \\ \text{subject to } & \nabla c_i(x_k)^T p + c_i(x_k) = 0,\ i \in \mathcal{E} \\ & \nabla c_i(x_k)^T p + c_i(x_k) \geq 0,\ i \in \mathcal{I} \end{aligned}$$

18.3. Sequential Quadratic Programming

IQP finds a search direction and the set of active constraints at the same time.

EQP selects a working set of constraints and solves equality-constrained problems. Then, updates the working set according to rules depending on the Lagrange multipliers.

Theorem 18.1 (Robinson [267]).

Suppose that x^* is a local solution of (18.10) at which the KKT conditions are satisfied for some λ^*. Suppose, too, that the linear independence constraint qualification (LICQ) (Definition 12.4), the strict complementarity condition (Definition 12.5), and the second-order sufficient conditions (Theorem 12.6) hold at (x^*, λ^*). Then if (x_k, λ_k) is sufficiently close to (x^*, λ^*), there is a local solution of the subproblem (18.11) whose active set $\mathcal{A}_k$ is the same as the active set $\mathcal{A}(x^*)$ of the nonlinear program (18.10) at x^*.

18.3. Sequential Quadratic Programming

SQP method is globally convergent and has the rate of superlinear convergence under some suitable assumptions.

Maratos observed that the unit step-size is not reached even at a superlinearly convergent step. Thus, the rate of convergence of the SQP method is only linear. This phenomenon is called "Maratos effect"

18.3. Sequential Quadratic Programming

We should take care of inconsistent constraint linearizations

Consider the inequalities $x \leq 1$ and $x^2 \geq 4$, which is feasible for $x \leq -2$. The linearization around $x_k = 1$ gives $-p \geq 0$ and $2p - 3 \geq 0$ which is infeasible.

Solution reformulation of the problem as the ℓ_1 penalty

$$f(x_k) + \nabla f^T(x_k)p + \frac{1}{2}p^T\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)p +$$
$$\mu \sum_{i \in \mathcal{E}} |\nabla c_i(x_k)^T p + c_i(x_k)| + \mu \sum_{i \in \mathcal{I}} [\nabla c_i(x_k)^T p + c_i(x_k)]^-$$

18.3. Sequential Quadratic Programming

with a smooth reformulation

$$
\begin{aligned}
\min_{p,r,s,t} \quad & f(x_k) + \nabla f^T(x_k)p + \frac{1}{2}p^T\nabla^2_{xx}\mathcal{L}(x_k,\lambda_k)p \\
& + \mu\sum_{i\in\mathcal{E}}(r_i + s_i) + \mu\sum_{i\in\mathcal{I}} t_i \\
\text{subject to} \quad & \nabla c_i(x_k)^T p + c_i(x_k) = r_i - s_i, \quad i \in \mathcal{E} \\
& \nabla c_i(x_k)^T p + c_i(x_k) \geq -t_i, \quad i \in I \\
& r, s, t \geq 0
\end{aligned}
$$

which is always feasible. If μ large enough and $v_i^* = w_i^* = t_i^* = 0$, x^* is a solution to the original subproblem, otherwise gives a measure of infeasibility (leads to Sℓ_1QP method)

18.3. Sequential Quadratic Programming

We still have two problems: 1) how to efficiently compute the Hessian matrix of the Lagrange function; 2) how to guarantee the positive definiteness of the Hessian matrix.

Difficult to compute $\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$. Idea use a quasi Newton approximation B_k (recall: matches the gradient of two past iterations)

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})$$

BFGS or SR1 update rules can be used.

Works nice if $\nabla^2_{xx}\mathcal{L} > 0$. If not $s_k^T y_k > 0$ (curvature condition) may not be satisfied and may behave poorly.

18.3. Sequential Quadratic Programming

Going around the problem. Skip update if $s_k^T y_k \leq s_k^T B_k s_k$ (not so good). Use a damped BFGS updating.

Procedure 18.2 (Damped BFGS Updating).
Given: symmetric and positive definite matrix B_k;
Define s_k and y_k as in (18.13) and set

$$r_k = \theta_k y_k + (1 - \theta_k) B_k s_k,$$

where the scalar θ_k is defined as

$$\theta_k = \begin{cases} 1 & \text{if } s_k^T y_k \geq 0.2 s_k^T B_k s_k, \\ (0.8 s_k^T B_k s_k)/(s_k^T B_k s_k - s_k^T y_k) & \text{if } s_k^T y_k < 0.2 s_k^T B_k s_k; \end{cases}$$

Update B_k as follows:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{r_k r_k^T}{s_k^T r_k}.$$

This is the same as BFGS with r_k replaced by y_k. If $\theta \neq 1$, update is $s_k^T r_k = 0.2 s_k^T B_k s_k > 0$

18.3. Sequential Quadratic Programming

The term $\nabla^2_{xx}\mathcal{L}_k$ affects only the part of p in the null space of $A(x_k)$

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}_k & -A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} p_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -\nabla f_k \\ -c_k \end{bmatrix}$$

Idea: Quasi-Newton approx. of $\nabla^2_{xx}\mathcal{L}_k$ that only affect that part!

Start by defining nonsingular $[Y_k|Z_k]$ with $A_k Z_k = 0$ and write solutions as $p_k = Y_k p_y + Z_k p_z$, giving:

$$(A_k Y_k) p_y = -c_k$$
$$(Z_k^T \nabla^2_{xx}\mathcal{L}_k Z_k) p_z = -Z_k^T \nabla^2_{xx}\mathcal{L}_k Y_k p_y - Z_k^T \nabla f_k$$

Make some simplifications to avoid computing $\nabla^2_{xx}\mathcal{L}_k$

18.3. Sequential Quadratic Programming

The multipliers are given from first row as

$$(A_k Y_k)^T \lambda_{k+1} = Y_k^T (\nabla f_k + \nabla^2_{xx} \mathcal{L}_k p_k)$$

Simp. 1: Neglect $\nabla^2_{xx} \mathcal{L}_k p_k$ (small close to solution) and use $Y_k = A_k^T$.

$$\hat{\lambda}_{k+1} = (A_k A_k^T)^{-1} A_k \nabla f_k = \arg\min_{\lambda} \|\nabla_x \mathcal{L}(x_k, \lambda)\|_2^2$$

It pushes λ to satisfy the f.o. KKT in a least squares sense (*least-squares multipliers*)

18.3. Sequential Quadratic Programming

Simp. 2: neglect the $n-m \times m$ cross term $Z_k^T \nabla_{xx}^2 \mathcal{L}_k Y_k p_y$ (p_y converges to zero faster than p_z, depends only on the active set)

the remaining $n-m \times n-m$ term $Z_k^T \nabla_{xx}^2 \mathcal{L}_k Z_k p_z$ simplifies as (using Taylor approximation)

$$Z_k^T \nabla_{xx}^2 \mathcal{L}_k Z_k \alpha_k p_z \approx Z_k^T [\nabla_x \mathcal{L}(x_k + \alpha_k p_k, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})]$$

Finally, we get the reduced quasi Newton approximation with secant $M_{k+1} s_k = y_k$ and

$$s_k = \alpha_k p_z, \quad y_k = Z_k^T [\nabla_x \mathcal{L}(x_k + \alpha_k p_k, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})]$$

and use BFGS or SR1 update rules.

18.4. References

[1] http://che.utexas.edu/course/che356/Chapter_8.ppt

[2] http://users.isy.liu.se/en/rt/andrecb/NumOpt/chap18_1-3.pdf

[3] LI-PING ZHANG, SLP & SQP Methods

[4] http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/frank-wolfe.pdf