Convex Optimization II

Lecture 18: Sequential Convex Programming: A Method for Non-convex Optimization Problems

Hamed Shah-Mansouri

Department of Electrical Engineering

Sharif University of Technology

1400-2

MOTIVATIONS

For convex optimization problems, in general, we can *easily* obtain a local optima which is the same as the *global* solution.

For non-convex optimization problems, we have to give up one. We can consider two approaches:

- **Local optimization methods**: Fast but sub-optimal
- **Global optimization methods**: Often slow, but attain the global solution.

[1] John Duchi, "Lecture Notes on Sequential Convex Programming", [Online] `https://web.stanford.edu/class/ee364b/lectures/seq_notes.pdf`

Thanks to Professor John Duchi, Stanford University, for the slides.

Sequential Convex Programming

Sequential Convex Programming (SCP): A local optimization method for non–convex problems that leverages convex optimization.

- Handle the convex portions of the problem exactly and efficiently
- Model the non–convex portions of the problem by convex functions that are (at least locally) accurate.

SCP is a heuristic method,

- may fail to find an optimal (or even feasible) point,
- may depend on the starting point,

but often finds good solutions.

Sequential Convex Programming

An iterative approach

- At iteration k, for any non-convex function f, form a model $\widehat{f}$ that is good enough near the current iterate $x^{(k)}$.
- Then, minimize that model or a regularized version of it, and repeat.

Sequential Convex Programming

Consider the (potentially non–convex) problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \text{for } i = 1, \ldots, m \\ & h_j(x) = 0, \text{for } j = 1, \ldots, p \\ & x \in \mathbb{R}^n. \end{array}$$

Here, the functions f_0 and f_i are (possibly) non–convex, and the functions h_j may be non-affine.

SCP: iterate by maintaining an estimate of the solution $x^{(k)}$ and a convex trust region, denoted by $\mathcal{T}^{(k)} \subset \mathbb{R}^n$, over which we trust our solutions and models.

Sequential Convex Programming

The generic SCP strategy then forms a

- convex approximation $\widehat{f}_i$ of the functions f_i over the trust region $\mathcal{T}^{(k)}$
- affine approximation $\widehat{h}_j$ of the functions h_j over the trust region $\mathcal{T}^{(k)}$

SCP iterations:

$$
\begin{array}{rll}
x^{(k+1)} = & \text{arg min} & \widehat{f}_0(x) \\
& \text{subject to} & \widehat{f}_i(x) \leq 0, \text{for } i = 1, \ldots, m \\
& & \widehat{h}_j(x) = 0, \text{for } j = 1, \ldots, p \\
& & x \in \mathcal{T}^{(k)}.
\end{array}
$$

Trust Region

ℓ_2-norm Ball

$$\mathcal{T}^{(k)} = \{x \in \mathcal{R}^n \mid ||x - x^{(k)}||_2 \leq \rho\}.$$

Box

$$\mathcal{T}^{(k)} = \{x \in \mathcal{R}^n \mid |x_i - x_i^{(k)}| \leq \rho_i,\ i = 1, \ldots, n\}.$$

In the latter, if x_i appears only in convex objectives and constraints, we can set $\rho_i = \infty$.

Affine and Convex Approximations

$\widehat{f}$ either takes an affine (first–order) Taylor approximation

$$\widehat{f}(x) = f(x^{(k)}) + \Delta f(x^{(k)})^T (x - x^{(k)})$$

or the convex part of the second order Taylor expansion

$$\widehat{f}(x) = f(x^{(k)}) + \Delta f(x^{(k)})^T (x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T P (x - x^{(k)})$$

where P is the positive semidefinite part of the Hessian $\Delta^2 f(x^{(k)})$. That is, if

$$\Delta^2 f(x^{(k)}) = U \Lambda U^T \quad \Longrightarrow \quad P = U[\Lambda]_+ U^T.$$

A Numerical Example

Non-convex Quadratic Program

$$\begin{array}{ll} \text{minimize} & f(x) = \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & ||x||_\infty \leq 1. \end{array}$$

where P is symmetric but not positive semindefinite.

Second–order Taylor (convex) approximation

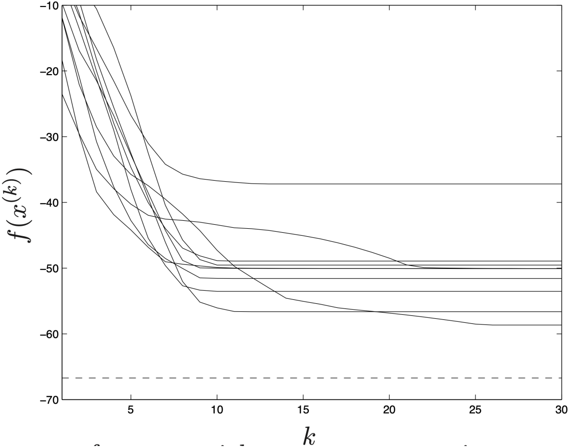
$$\widehat{f}(x) = f(x^{(k)}) + (Px^{(k)} + q)^T (x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T P_+ (x - x^{(k)})$$

Trust Region

$$\mathcal{T}^{(k)} = \{x \in \mathcal{R}^n \mid ||x - x^{(k)}||_\infty \leq \rho\}.$$

A NUMERICAL EXAMPLE

Example with $n = 20$ and $\rho = 0.2$, and different initial points.



Sequential Convex Programming Issues

Each iteration is, in principle, easy to solve as it is a convex problem. However, numerous issues arise:

Infeasibility: How to decide when to accept a step by taking into account:

- constraint violations,
- trade–off between feasibility of constraints and quality of the objective.

Trust region size: How large trust region should be.

- ρ too large, poor approximations
- ρ too small, more accurate approximation, but slow.

Infeasibility Issue via Penalization

Workaround: Assign penalties to constraint violations rather than to directly enforce the constraints.

Replace the original problem with a penalized approximation

$$\text{minimize} \quad \phi(x) = f_0(x) + \lambda \left(\sum_{i=1}^{m} [f_i(x)]_+ + \sum_{j=1}^{p} |h_j(x)| \right)$$

This approach can often allow progresses that would otherwise be impossible.

For λ large enough, this turns into exact penalization. The solutions will satisfy the constraints.

Infeasibility Issue via Penalization

So, we will replace the update in SCP by:

$$\begin{array}{rll} \tilde{x} = & \arg\min & \widehat{\phi}(x) = \widehat{f}_0(x) + \lambda \left(\sum_{i=1}^{m} [\widehat{f}_i(x)]_+ + \sum_{j=1}^{p} |\widehat{h}_j(x)| \right) \\ & \text{subject to} & x \in \mathcal{T}^{(k)}. \end{array}$$

If this provides a good solution, we accept it and set

$$x^{(k+1)} = \tilde{x}$$

Trust Region Size Issue

Adjust the radius of the trust region considering how good is the approximation.

The trust region is:

$$\mathcal{T}^{(k)} = \{x \in \mathcal{R}^n \mid ||x - x^{(k)}||_2 \leq \rho\}.$$

Predicted Decrease:

$$\widehat{\delta} = \phi\left(x^{(k)}\right) - \widehat{\phi}(\tilde{x}).$$

Actual Decrease:

$$\delta = \phi\left(x^{(k)}\right) - \phi(\tilde{x}).$$

Trust Region Size Issue

Let $\alpha \in (0, \frac{1}{2})$ and $\beta^{\text{succ}} > 1$ and $\beta^{\text{fail}} < 1$.

Algorithm

- If $\delta \geq \alpha\widehat{\delta}$ (i.e., sufficient decrease),
  1. accept the solution,
  2. set $x^{(k+1)} = \tilde{x}$,
  3. enlarge the trust region by $\rho^{(k+1)} = \beta^{\text{succ}}\rho^{(k)}$.
- Otherwise,
  1. reject the step,
  2. shrink the trust region by $\rho^{(k+1)} = \beta^{\text{fail}}\rho^{(k)}$,
  3. obtain $\tilde{x}$ again.