COMP 9602: Convex Optimization

Methods for Nonconvex Optimization

Dr. C Wu

Department of Computer Science
The University of Hong Kong

Methods for nonconvex optimization problems

- Convex optimization algorithms are in general
  - global (find global minimum)
  - fast (run in polynomial-time)

- For general nonconvex problems, we have to give up one
  - local optimization methods are fast, but may not find global minimum (and even when they do, cannot certify it)
  - global optimization methods find global minimum (and certify it), but are often slow

Branch and bound

- Methods for global optimization for nonconvex problems
- Basic idea
  - partition feasible set into convex sets, and find lower/upper bounds for each
  - maintain global lower and upper bounds; quit if they are close enough to each other
  - else, refine partition and repeat
- Often slow; exponential worst-case performance
- but (with luck) can (sometimes) work well

Branch and bound algorithm for unconstrained optimization

- Problem: find global minimum of function $f : \mathbf{R}^m \to \mathbf{R}$, over a m-dimensional rectangle $\mathcal{Q}_{\text{init}}$, to within some preset accuracy ϵ

- Algorithm sketch:

1. compute lower and upper bounds on $f^\star$
   - set $L_1 = \Phi_{\text{lb}}(\mathcal{Q}_{\text{init}})$ and $U_1 = \Phi_{\text{ub}}(\mathcal{Q}_{\text{init}})$
   - terminate if $U_1 - L_1 \leq \epsilon$
2. partition (split) $\mathcal{Q}_{\text{init}}$ into two rectangles $\mathcal{Q}_{\text{init}} = \mathcal{Q}_1 \cup \mathcal{Q}_2$
3. compute $\Phi_{\text{lb}}(\mathcal{Q}_i)$ and $\Phi_{\text{ub}}(\mathcal{Q}_i)$, $i = 1, 2$
4. update lower and upper bounds on $f^\star$
   - update lower bound: $L_2 = \min\{\Phi_{\text{lb}}(\mathcal{Q}_1), \Phi_{\text{lb}}(\mathcal{Q}_2)\}$
   - update upper bound: $U_2 = \min\{\Phi_{\text{ub}}(\mathcal{Q}_1), \Phi_{\text{ub}}(\mathcal{Q}_2)\}$
   - terminate if $U_2 - L_2 \leq \epsilon$
5. refine partition, by splitting $\mathcal{Q}_1$ or $\mathcal{Q}_2$, and repeat steps 3 and 4

Local search

- A heuristic method to solve computationally hard problems
  - moves from solution to solution in the space of candidate solutions (i.e., search space) by applying local changes, until a solution deemed optimal is found or a bound on the number of steps/time taken is reached
- Problems that local search has been applied for
  - minimum vertex cover problem
  - travelling salesman problem
  - boolean satisfiability problem

Local search

- A neighborhood of a solution needs to be defined in the search space
  - minimum vertex cover problem

    neighborhood of a vertex cover c: contains all vertex covers differing from c only by one node
  - travelling salesman problem

    k-change neighbourhood for a tour f: contains all tours that can be obtained from f by removing k edges and replacing them with k edges

Key elements in a local search algorithm

- How to choose a good neighbourhood for the problem
  - guided by intuition
  - very little theory available as a guide

- Which solution in the neighborhood to move to
  - decided using only information in the neighborhood
  - can get stuck in a local optimal point

    tackled by starting local search at different initial points
    or using more complex approaches, e.g., iterate local search, simulated annealing

- How to obtain the starting feasible solution(s)
  - how many starting points to try and how to distribute them

Sequential convex programming (SCP)

- A local optimization method for nonconvex problems based on solving a sequence of convex problems
- SCP is a heuristic
  - may fail to find optimal (even feasible) point
  - results often depend on the starting point(s)
- SCP often works well, i.e., finds a feasible point with good, if not optimal, objective value

Problem

we consider nonconvex problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad j = 1, \ldots, p \end{array}$$

with variable $x \in \mathbf{R}^n$

- f_0 and f_i (possibly) nonconvex
- h_i (possibly) non-affine

Basic idea of SCP

- maintain estimate of solution $x^{(k)}$, and convex **trust region** $\mathcal{T}^{(k)} \subset \mathbf{R}^n$
- form convex approximation $\hat{f}_i$ of f_i over trust region $\mathcal{T}^{(k)}$
- form affine approximation $\hat{h}_i$ of h_i over trust region $\mathcal{T}^{(k)}$
- $x^{(k+1)}$ is optimal point for approximate convex problem

$$
\begin{array}{ll}
\text{minimize} & \hat{f}_0(x) \\
\text{subject to} & \hat{f}_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \hat{h}_i(x) = 0, \quad i = 1, \ldots, p \\
& x \in \mathcal{T}^{(k)}
\end{array}
$$

Trust region

- typical trust region is box around current point:

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

- if x_i appears only in convex inequalities and affine equalities, can take $\rho_i = \infty$

Affine and convex approximation

- For differentiable functions
  - Use first-order Taylor expansion for the affine approximation

$$\hat{f}(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T (x - x^{(k)})$$

  - Use (convex part of) second-order Taylor expansion for the convex approximation

$$\hat{f}(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T (x - x^{(k)}) + (1/2)(x - x^{(k)})^T P (x - x^{(k)})$$

$$P = \left(\nabla^2 f(x^{(k)})\right)_+, \text{ PSD part of Hessian}$$

Affine and convex approximation

- For nondifferentiable and differentiable functions
  - particle method

    – choose points $z_1, \ldots, z_K \in \mathcal{T}^{(k)}$
    ($e.g.$, all vertices, some vertices, grid, random, . . .)
    – evaluate $y_i = f(z_i)$
    – fit data (z_i, y_i) with convex (affine) function
    (using convex optimization)

- Reference
  - branchbound_notes.pdf, scp_notes.pdf (reference 9 on Moodle)
  - Chapter 18, 19, C. Papadimitriou, K. Steiglitz, Combinational Optimization: Algorithms and Complexity

- Acknowledgement
  - Some materials are extracted from the slides created by Prof. Stephen Boyd for his course EE364b in Stanford University