Algorithms and Applications of Linear Programming —- Final Project for "Convex Optimization"

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

September 29, 2017

1 Semismooth Newton Algorithms for the standard form LP

Consider the standard form of LP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ s.t. \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{1.1}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ are given. The dual problem is

$$\begin{aligned} \max_{y \in \mathbb{R}^m, s \in \mathbb{R}^n} \quad & b^T y \\ s.t. \quad & A^T y + s = c, \\ & s \geq 0. \end{aligned} \tag{1.2}$$

1. Write down and implement an augmented Lagrangian method for solving (1.2).
   (a) Write down an augmented Lagrangian method for solving the dual problem (1.2), where the variable s is eliminated (i.e., the variable s should not appear in the update of the algorithm).
   (b) Method 1: Apply a gradient-type method to minimize each augmented Lagrangian function. It can be a method from Homework 5.
   (c) Method 2: Write down a semi-smooth Newton method for minimizing each augmented Lagrangian function. A reference is:
   Zhao, Xin-Yuan, Defeng Sun, and Kim-Chuan Toh. "A Newton-CG augmented Lagrangian method for semidefinite programming." SIAM Journal on Optimization 20.4 (2010): 1737-1765.
   `http://epubs.siam.org/doi/abs/10.1137/080718206`.
2. Semi-smooth Newton method based on solving a fixed-point equation.
   (a) Write down and implement the DRS for (1.1) and ADMM for the dual problem of (1.2).
   (b) Derive the explicit relationship between the variables of DRS and ADMM mentioned above.

(c) Write down and implement a regularized semi-smooth Newton method for solving (1.1). References are sections 3 and 4 in

- Yongfeng Li, Zaiwen Wen, Chao Yang, Yaxiang Yuan, A Semi-smooth Newton Method for Solving Semidefinite Programs in Electronic Structure Calculations,
  `https://arxiv.org/abs/1708.08048`

3. Requirement:

(a) The interface of each method should be written in the following format

```
[x, out] = method_name(c, A, b, opts, x0);
```

Here, c, A and b are given data, opts is a struct which stores the options of the algorithm, out is a struct which saves all other output information. The parameter x0 is an optional given input initial solution. In other words, x0 is not necessarily required as an input. The programming language can be Matlab or Python.

(b) Test problems:

- Random data:

```
n = 100;
m = 20;
A  = rand(m,n);
xs = full(abs(sprandn(n,1,m/n)));
b  = A*xs;
y  = randn(m,1);
s  = rand(n,1).*(xs==0);
c  = A'*y + s;
```

- Netlib test problems. A matlab version of these data can be found at:
  `http://bicmr.pku.edu.cn/~wenzw/code/MPS-presolve-mat.zip`
  Note that the problems in the Netlib may not be in the standard form (1.1). They can be as general as

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & c^T x \\
s.t. \quad & b_l \le Ax \le b_u, \\
& t_l \le x \le t_u.
\end{aligned} \tag{1.3}
$$

(c) Compare the efficiency (cpu time) and accuracy (checking optimality condition) with the LP solvers in Mosek or Gurobi.

(d) Prepare a report including

- detailed answers to each question
- numerical results and their iterpretation

(e) Pack all of your codes in one file named as "proj2-name-ID.zip" and send it to TA:
pkuopt@163.com

(f) If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

2 Algorithms for large-scale Optimal Transport

Consider the standard form of LP

$$
\begin{aligned}
\min_{\pi \in \mathbb{R}^{m \times n}} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \pi_{ij} \\
s.t. \quad & \sum_{j=1}^{n} \pi_{ij} = \mu_i, \quad \forall i = 1, \dots, m, \\
& \sum_{i=1}^{m} \pi_{ij} = \nu_i, \quad \forall j = 1, \dots, n, \\
& \pi_{ij} \geq 0.
\end{aligned} \tag{2.1}
$$

1. Solve (2.1) by calling mosek and gurobi directly in Matlab or python. The package "CVX" is not allowed to use here. Compare the performance between the simplex methods and interior point methods.
2. Write down and implement a first-order method, for example, the alternating direction method of multipliers.
3. Read the reference:
   Jörn Schrieber, Dominic Schuhmacher, Carsten Gottschlich, DOTmark A Benchmark for Discrete Optimal Transport.
   Then write down and implement one of the following methods:
   - transportation simplex
   - shortlist method
   - shielding neighorhood method
   - AHA method
4. Read the reference:
   Samuel Gerber, Mauro Maggioni, Multiscale Strategies for Computing Optimal Transport.
   Then write down and implement the multiscale method.
5. Requirement:
   (a) Test problems:
   - Generate some random data c, μ and nu.
   - Find or construct the data sets in the references:
     - Jörn Schrieber, Dominic Schuhmacher, Carsten Gottschlich, DOTmark A Benchmark for Discrete Optimal Transport.
     - Samuel Gerber, Mauro Maggioni, Multiscale Strategies for Computing Optimal Transport.

   (b) Compare the efficiency (cpu time) and accuracy (checking optimality condition) of different methods.

   (c) Prepare a report including
   - detailed answers to each question
   - numerical results and their iterpretation

(d) Pack all of your codes in one file named as "proj2-name-ID.zip" and send it to TA: pkuopt@163.com

(e) If you get significant help from others on one routine, write down the source of references at the beginning of this routine.