ME 555: Distributed Optimization

Duke University

Spring 2015

Administrative

Course:	ME 555: Distributed Optimization (Spring 2015)
Instructor:	Soomin Lee (email: s.lee@duke.edu)
Time:	TuTh 3:05 - 4:20 pm
Location:	232 Hudson Hall
Office hours:	TBA
Website:	http://sites.duke.edu/me555_07_s2015

Prerequisites

- CEE 690: Introduction to Numerical Optimization or equivalent
- Consent of instructor

We will review necessary concepts, but I will assume you already have some knowledge on:

1. Multivariate Calculus
2. Real Analysis
3. Linear Algebra
4. Basic Numerical Methods: Gradient Descent, Newton Method

References

No textbook is required. There will be some reading assignments.

1. D. Bertsekas, Nonlinear Programming, Athena Scientific, 2nd Edition.
2. A. Ruszczynski, Nonlinear Optimization, Princeton University Press.
3. Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Springer.
4. D. Bertsekas and J. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Athena Scientific.
5. J. Nocedal and S. Wright, Numerical Optimization, Springer, 2nd Edition.
6. S. Boyd and L. Vandenverghe, Convex Optimization, Cambridge University Press.

Grading

- Homework (6 sets): 30%

The last three weeks will be devoted to presentations. Each student will give two presentations.

- Paper Presentation: 20%
  - Papers to be announced by instructor
  - Peer evaluated
- Final Project: 50%
  - Any theoretical topics or applications relevant to optimization
  - Individual project encouraged

Final Project: Time Line

- Proposal (1 page) by January 27th, in class
- Interim Report (1 page) by March 5th, in class
- Presentation (20-30 minutes) on April 7th, 9th and 14th
- Final Report (5-6 page, double column) by 5pm, April 24th

Optimization History

- Optimization Theory and Analysis have been studied for a long time, mostly by mathematicians
- Until late 1980s:
  - Algorithms mainly focused on solving Linear Problems
    - Simplex Algorithm for linear programming (Dantzig, 1947)
    - Ellipsoid Method (Shor, 1970)
    - Interior-Point Methods for linear programming (Karmarkar, 1984)
  - Applications mostly in operations research and few in engineering
- After late 1980s:
  - A new interest in optimization emerges in various fields
    - Automatic Control Systems
    - Estimation, Signal and Image Processing
    - Communication and Data Networks
    - Data Analysis and Modeling
    - Statistics and Finance

Newly Emerging Interest in Optimization

- The end of Moore's Law promoted parallel computing
- Networked systems are now pervasive in practice
  - Communication networks
  - Social networks
  - Power grid
  - Transport grid
  - Decentralized computing networks
  - Military applications

Distributed optimization algorithms and analysis
are now more important than ever!

Mathematical Formulation of Optimization Problem

$$\min_{x} \; f(x)$$
$$\text{subject to } x \in X$$

where

$$X = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, \text{ for } i = 1, \ldots, m\}.$$

- $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function* (or cost function).
- $x \in \mathbb{R}^n$ are the *decision variables*.
- $X \subseteq \mathbb{R}^n$ is the *constraint set* (or feasible set).
- $g_i : \mathbb{R}^n \to \mathbb{R}$ are *constraint functions*.

Goal: We would like to find a solution $x^* \in X$ such that

$$f(x^*) \leq f(x) \quad \forall x \in X$$

Mathematical Formulation of Optimization Problem

$$\min_{x} \; f(x)$$

subject to $x \in X$

- We say x is a *feasible point* if $x \in X$.
- A feasible point $x^* \in X$ is an *optimal point* (or optimum) if

$$f(x^*) \leq f(x) \quad \forall x \in X$$

- f^* is the *optimal value*

$$f^* \triangleq f(x^*)$$

- X^* is the *optimal set*

$$X^* = \{x \in X \mid f(x) = f^*\}$$

Note: X^* is not always nonempty. Therefore, whenever you encounter an optimization problem, the first thing to do is to check the existence of the solution.

Some Examples

Communication Networks

- Variables: communication rates for users
- Constraints: link capacities
- Objective: user cost

Portfolio Optimization

- Variables: amounts invested in different assets
- Constraints: available budget, maximum/minimum investment per asset, minimum return, time constraints
- Objective: overall risk or return variance

Data Fitting

- Variables: model parameters
- Constraints: prior information, parameter limits
- Objective: measure of misfit or prediction error

Classification of Optimization Problems

1. Constrained vs. Unconstrained

$$\min_{x} \; f(x)$$
$$\text{subject to } x \in X$$

- If $X = \mathbb{R}^n$, unconstrained.
- If $X \subseteq \mathbb{R}^n$ and $X \neq \mathbb{R}^n$, constrained.

- Constrained problems are usually much harder to solve.

Classification of Optimization Problems

2. Linear $\subseteq$ Convex $\subseteq$ Nonlinear

$$\min_x \; f(x)$$
$$\text{subject to } x \in X$$

where

$$X = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, \text{ for } i = 1, \ldots, m\}.$$

- If f and g_i are linear (or X is polyhedral), the problem is called *Linear Programming*.
- If f and g_i are convex (or X is a convex set), the problem is called *Convex Programming*.
- If f and g_i are nonlinear, the problem is called *Nonlinear Programming*.

Classification of Optimization Problems

3. Continuous vs. Discrete (Mixed Integer)

$$\min_{x} \; f(x)$$
$$\text{subject to } x \in X$$

- The problem is *continuous* if $x \in \mathbb{R}^n$.
- The problem is *discrete* (or mixed integer) if $x = [y^T \; z^T]^T \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}$.

  Here, $n = n_1 + n_2$.
- Discrete problems are usually much harder to solve.

Classification of Optimization Problems

4. Deterministic vs. Stochastic

$$\min_{x} \; \mathsf{E}_{\xi}[f(x, \xi)]$$
$$\text{subject to } x \in X$$

- The problem is *deterministic* if ξ is deterministic.
- The problem is *stochastic* (or uncertain) if ξ is a random variable following some known or unknown distribution.

Classification of Optimization Problems

5. Centralized vs. Distributed

$$\min_{x} \; f(x)$$
$$\text{subject to } x \in X$$

- The problem is *centralized* if a single computing unit (or decision maker) solves the problem.
- The problem is *distributed* if multiple computing units (where each unit has either partial or full information about the problem) cooperatively solve the problem together.

Motivation of Distributed Optimization

1. Computational Issue

- Optimization problem is large-scale
  - Either n or m (or both) is really huge
- Limited time and/or memory
- For some problems, complexity grows more than linearly
- Examples:
  - Weather Prediction
  - Image Processing
  - Machine Learning with Big Data

Necessity of problem decomposition and parallel processing

Motivation of Distributed Optimization

Decomposition: Ideal Case

$$\min_{x_1, x_2} \; f_1(x_1) + f_2(x_2)$$

where $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$

- Let us say $n_1 = 10^9$ and $n_2 = 10^9$.
- Our system has 8GB memory and uses double-precision floating point format.
- This problem cannot be saved or processed in a single system.

- Let system 1 solve the subproblem $\min_{x_1 \in \mathbb{R}^{n_1}} f_1(x_1)$ and system 2 solve the subproblem $\min_{x_2 \in \mathbb{R}^{n_2}} f_2(x_2)$.
- Our final solution is $[x_1^* \; x_2^*]$ and the optimal value is $f_1(x_1^*) + f_2(x_2^*)$.
- But life is not that easy.

Motivation of Distributed Optimization

Decomposition: More Interesting Case

$$\min_{x_1, x_2, y} \; f_1(x_1, y) + f_2(x_2, y)$$

where $x_1 \in \mathbb{R}^{n_1},\; x_2 \in \mathbb{R}^{n_2}$ and $y \in \mathbb{R}^{n_3}$

- We can assume $n_1, n_2 \gg n_3$
- Majority of distributed optimization methods is about how to enforce the consistency in y across many decision makers.
- We are going to learn those methods in this class.
- If the problem is convex, we have more options for handling the coupled variables.

Motivation of Distributed Optimization

2. **Structural Issue**

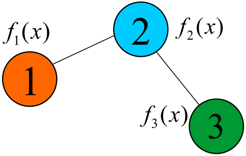
- Decision makers are geographically distributed
- Limited communication
- Examples:
  - P2P Networks, Ad-hoc Networks
  - Social Networks
  - Transport and Power Grid
  - Fleets of Robots

> The problem is naturally decomposed and
> its communication structure is already given.

Motivation of Distributed Optimization

A Distributed Problem in a Networked System

$$\min_{x \in \mathbb{R}^n} \; f(x) \triangleq f_1(x) + f_2(x) + f_3(x)$$



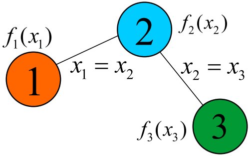
- $f_i(x)$ is the local objective of subsystem i.
- $f(x)$ is the system-wide objective.
- Hence, the subsystems are coupled through x.

Motivation of Distributed Optimization

An Equivalent Form

$$\min_{x_1, x_2, x_2} \quad f_1(x_1) + f_2(x_2) + f_3(x_3)$$

$$\text{s.t. } x_1 = x_2 \text{ and } x_2 = x_3$$



- Let each system i solve the subproblem $\min_{x_i} f_i(x_i)$.
- Subsystem 1 and 2 communicate and try to enforce $x_1 = x_2$.
- Subsystem 2 and 3 communicate and try to enforce $x_2 = x_3$.

- We are going to learn variants of such methods in this class.
- Again, if the problem is convex, we have more options.

List of Topics (Tentative)

1. Introduction and Motivation
2. Review of Basic Concepts from Real Analysis and Linear Algebra
3. Convex Sets and Convex Functions
4. Convex Problems and Optimality
5. Separation Theorems and Lagrangian Duality
6. Mathematical Decomposition
7. Proximal Algorithms and Augmented Lagrangian Methods
8. Recent Convex Optimization Algorithms for Big Data
9. Spectral Graph Theory / Subgradient Methods
10. Consensus Theorem
11. Subgradient Consensus Schemes

Goal of this course

This is a theory course with lots of mathematics!

- To provide you with working knowledge (basic terminology, principles and methodologies) of optimization
- To develop convergence proof techniques of optimization algorithms
- To study optimization algorithms that can be distributed across many decision makers
- To understand the information aggregation / propagation mechanism of geographically distributed systems
- To develop an ability to decompose large-scale problems
- To know which distributed algorithm is best suitable for your purpose
- To design new distributed algorithms or improve the efficiency of existing ones

Topics Not Covered

- Nonconvex, stochastic, discrete optimization problems / algorithms
- Optimization softwares and corresponding algebraic mathematical programming languages
- Parallel computation framework

  e.g. GPU, Hadoop, MapReduce, Spark, ...
- High level parallel computation languages

  e.g. Pig, Hive, Jaql, ...
- Instead, we will assume a computation unit can be anything
  - Personal computers or smartphones interconnected over TCP/IP
  - Unmanned vehicles communicating with each others
  - Sensing units in mobile sensor networks
  - CPU cores with shared memory