COM S 672: Advanced Topics in Computational Models of Learning – Optimization for Learning

Lecture Note 1: Course Info & Introduction

Jia (Kevin) Liu

Assistant Professor
Department of Computer Science
Iowa State University, Ames, Iowa, USA

Fall 2017
Course Info (1)

- **Instructor:** Jia (Kevin) Liu, Asst. Professor  
- **Office:** 209 Atanasoff Hall  
- **Email:** jialiu@iastate.edu  
- **Time:** TuTh 5:10pm – 6:25pm  
- **Location:** 3143 Pearson Hall  
- **Office Hour:** Wed 5–6pm or by appointment  
- **Website:**  
  http://web.cs.iastate.edu/~jialiu/teaching/COMS672_F17/  
  (Canvas will be used for announcements, discussions, and grade management)  

**Prerequisite:**
- Working knowledge of **Real Analysis, Probability, Linear Algebra**  
- Exposure to optimization, Com S 572/573/472/474 is a plus but not required
Course Info (2)

Grading Policy:

- **Homework (30%)**
  - Assigned biweekly (approximately)
  - May involve open-ended questions
  - Must be typeset using \LaTeX
  - Some problems could be challenging!

- **Midterm (30%)**

- **Final Project (40%)**
  - Could be individual or team of 2. Project proposal due soon after midterm
  - Project report due in the final exam week. Follow IEEE journal format
    (It could become a publication of yours! 😊)
  - 15-minute in-class presentation at the end of the semester. Final report due by
    the beginning of final exam week (Dec. 11)
  - Potential ideas of project topics (should contain something new & useful):
    - Nontrivial extension of the results introduced in class
    - Novel applications in your own research area
    - New theoretical analysis/insights of an existing algorithm
    - It is important that you justify its novelty!
Course Info (3)

Course Materials:

- No required textbook

- Lecture notes are developed based on:
  
  
  
  

- Important & trending papers in the field (Reading list will be provided)
Tentative Topics

- Fundamentals of Convex Analysis
  - Convexity, optimality conditions, duality, ...

- First-Order Methods
  - Gradient descent, momentum, Nesterov, conjugate gradient, mirror descent, ...

- Higher-Order Methods
  - Newton, quasi-Newton family, interior-point method, ...

- Sparse/Regularized Optimization
  - Compressed sensing, matrix completion, ...

- Augmented Lagrangian Methods
  - ADMM methods, proximal methods, coordinate descent, ...

- If time allows:
  - Robust Optimization
  - Non-Convex Optimization
  - Multi-Arm Bandits
Special Notes

- Advanced, research-oriented, but not seminar type of course
  - There will be assignments and a midterm exam

- **Goal:** Prepare & train students for theoretical research

- But will (briefly) mention relevant applications in ML:
  - Artificial intelligence
  - Big data analytics
  - Robotics
  - Natural language processing ...

- **Caveat:** Focus on theory & proofs, rather than “coding/programming”
  - No “one book fits all” ⇒ Many readings required
  - Will try to cover a wide range of major topics
  - Background materials will be introduced but at very fast pace
  - So, mathematical maturity is essential!
How to Best Prepare for the Lectures?

Read, read, read!

- Especially if you're unfamiliar with the background (e.g., linear algebra, probability, ...)
  - Class time is limited.
  - Can’t afford to go over the background materials

- Appendices in [BV] and [BSS] provide lots of math background

- You are welcome to ask questions in office hours

- But you can’t rely on me to teach you the background in this course
Mathematical Optimization

Mathematical optimization problem:

\[
\begin{align*}
\text{Minimize} \quad & f_0(x) \\
\text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m
\end{align*}
\]

- \( x = [x_1, \ldots, x_N]^T \in \mathbb{R}^N \): decision variables
- \( f_0 : \mathbb{R}^N \rightarrow \mathbb{R} \): objective function
- \( f_i : \mathbb{R}^N \rightarrow \mathbb{R}, i = 1, \ldots, m \): constraint functions

Solution or optimal point \( x^* \) has the smallest value of \( f_0 \) among all vectors that satisfy the constraints
Solving Optimization Problems

- General optimization problems
  - Very difficult to solve (NP-hard in general)
  - Often involve trade-offs: long computation time, may not find an optimal solution (approximation may be acceptable in practice)

- Exceptions: Problems with special structures
  - Linear programming problems
  - Convex optimization problems
  - Some non-convex optimization problems with strong-duality
Brief History of Optimization

Theory:

- Early foundations laid by many all-time great mathematicians (e.g., Newton, Gauss, Lagrange, Euler, Fermat, ...)
- Convex analysis 1900–1970 (Duality by von Neumann, KKT conditions...)

Algorithms

- 1947: simplex algorithm for linear programming (Dantzig)
- 1970s: ellipsoid method (Khachiyan 1979), 1st polynomial-time alg. for LP
- since 2000s: many methods for large-scale convex optimization

Applications

- before 1990: mostly in operations research, a few in engineering
- since 1990: many applications in engineering (control, signal processing, networking and communications, circuit design,...)
- since 2000s: machine learning
Applying Optimization Tools in Machine Learning

- Linear Regression
- Variable Selection & Compressed Sensing
- Support Vector Machine
- Logistic Regression (+ Regularization)
- Matrix Completion
- Deep neural network training
- Reinforcement learning
- ...
Example 1: Linear Regression

Minimize $\beta \quad \| y - X\beta \|^2_2$

- Given data samples: $\{(x_i, y_i), i = 1, \ldots, m\}$, where $x_i \in \mathbb{R}^n$, $\forall i$
- Find a linear estimator: $y = \beta^T x$, so that “error” is small in some sense
- Let $X \triangleq [x_1, \ldots, x_n]^\top \in \mathbb{R}^{m \times n}$, $y \triangleq [y_1, \ldots, y_m]^\top \in \mathbb{R}^m$
- Linear algebra for $\| \cdot \|_2$: $\beta^* = (X^\top X)^{-1}X^\top y$ (analytical solution)
- Computation time proportional to $n^2m$ (less if structured)
- Stochastic gradient if $m, n$ are large
Example 2: Support Vector Machine (SVM)

- Given data samples: \( \{(x_i, y_i), i = 1, \ldots, m\} \)
  - \( x_i \in \mathbb{R}^n \) called “feature vectors”, \( \forall i \)
  - \( y_i \in \{-1, +1\} \) are “labels”

- Linear classifier: \( f(x) = \text{sgn}(w^T x + b) \)
  - \( w \in \mathbb{R}^n \): weight vector for features
  - \( b \in \mathbb{R} \): Some “bias”

Goal: To find a pair \((w, b)\) to minimize a weighted sum such that
- Minimize classification error on training samples
- Robust to random noise in the training samples

Minimize \( w, b, \epsilon \)
\[
\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \epsilon_i
\]
subject to \( y_i (w^T x_i + b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0, \quad i = 1, \ldots, m \)
Optimization Algorithms for SVM

- Coordinate Descent (Platt, 1999; Chang and Lin, 2011)
- Stochastic gradient (Bottou and LeCun, 2004; Shalev-Shwartz et al., 2007)
- Higher-order methods (interior-point) (Ferris and Munson, 2002; Fine and Scheinberg, 2001); (on reduced space) (Joachims, 1999)
- Shrink Algorithms (Duchi and Singer, 2009; Xiao, 2010)
- Stochastic gradient + shrink + higher-order (Lee and Wright, 2012)
Example 3: Compressed Sensing

Interested in solving **undetermined** systems of linear equations:

- Estimate $\mathbf{x} \in \mathbb{R}^n$ from linear measurements $\mathbf{b} = \mathbf{A} \mathbf{x} \in \mathbb{R}^m$, where $m \ll n$.
- Seems to be hopelessly ill-posed, since more unknowns than equations...
- Or does it?
A Little History of Compressive Sensing (CS)

- Name coined by David Donoho
- Pioneered by Donoho and Candès, Tao and Romberg in 2004
Sensing and Signal Recovery

Conventional paradigm of data acquisition: Acquire then compress

Q: Why compression works?
A: Quite often, there’s only marginal loss in “quality” between the raw data and its compression form.

Q: But still, why marginal loss?
Sparse Representation

- **Sparsity:** Many real world data admit sparse representation. The signal $s \in \mathbb{C}^n$ is sparse in a basis $\Phi \in \mathbb{C}^{n \times n}$ if

  $$s = \Phi x \quad \text{and} \quad x \in \mathbb{R}^n$$

  only has **very few non-zero elements**

- For example, images are sparse in the wavelet domain

- The # of large coefficients in the wavelet domain is small $\Rightarrow$ compression
Compressed Sensing: Compression on the Fly!

Q: Could we directly compress data and then reconstruct?

\[ y_i = \langle a_i, x \rangle, \quad i = 1, \ldots, m \]

**Goal:** To learn (recover) \( x \)'s value through some given (noisy) samples \( y_i \)?

- Mathematically, this gives rise to an underdetermined system of equations, where the signal of interests is **sparse**
Sparse Recovery

In optimization, CS can be written in the form of:

$$\min_{x \in \mathbb{R}^n} \phi_\gamma(x) \triangleq f(y, \Phi; x) + \gamma \|x\|_1$$

In machine learning context, questions of interests include:

- How to design the measurement/sampling matrix $\Phi$?
- What are the efficient algorithms to search for $x$?
- Are they stable under noisy inputs?
- How many measurements/samples are necessary/sufficient (i.e., size of $y$)?

**Insight:** Turns out $m = \Omega(\log(n))$ random samples will suffice
Some Optimization Algorithms for Compressed Sensing

- Shrink algorithms (for $l_1$ term) (Wright et al., 2009)
- Accelerated gradient (Beck and Teboulle, 2009b)
- ADMM (Zhang et al., 2010)
- Higher-order: Reduced inexact Newton (Wen et al., 2010); Interior-point (Fountoulakis and Gondzio, 2013)
Example 4: Matrix Completion – The Netflix Problem

In 2006, Netflix offered $1 million prize to improve movie rating prediction

- How to estimate the missing ratings?

About a million users, and 25,000 movies, with sparsely sampled ratings

- In essence, a low-rank matrix completion problem
Low-Rank Matrix Completion

- **Completion Problem:** Consider $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ to represent Netflix data, we may model it through factorization:

  \[
  \begin{bmatrix}
  \times & \times & \times \\
  \times & \times & \times \\
  \times & \times & \times \\
  \end{bmatrix}
  \overset{\cong}{\sim}
  \begin{bmatrix}
  \text{movies} \\
  \text{users} \\
  \end{bmatrix}
  \]

  \[\Rightarrow \text{rank}.\]

- In other words, the rank $r$ of $\mathbf{M}$ is much smaller than its dimension $r \ll \min\{n_1, n_2\}$
Low-Rank Matrix Completion

In optimization, the low-rank matrix completion problem can be written as:

\[
\begin{align*}
\text{Minimize} & \quad \text{rank}(X) \\
\text{subject to} & \quad (X)_{ij} = (M)_{ij}, \quad \forall i, j \in \text{observed entries}
\end{align*}
\]

In machine learning context, questions of interests include:

- What are the efficient algorithms to search for \( X \)?
- Are they stable under noisy inputs and outliers?
- How many samples are necessary/sufficient (i.e., size of \( (M)_{i,j} \))? 

Insight: Turns out \( m = \Omega(r \max\{n_1, n_2\} \log^2(\max\{n_1, n_2\})) \) samples will suffice
Some Optimization Algorithms for Matrix Completion

- (Block) Coordinate Descent (Wen et al., 2012)
- Shrink (Cai et al., 2010a; Lee et al., 2010)
- Stochastic Gradient (Lee et al., 2010)
Next Class...

We will start from some basic convex analysis.