ECE599/CS519: Convex Optimization

Lecture 1 - Introduction

Xiao Fu

School of Electrical Engineering and Computer Science
Oregon State University
E-mail: xiao.fu@oregonstate.edu
Course Information

• **Prerequisites:** Comfortable with Linear Algebra, Basic Programming Skills with Matlab or Python

• **Grading:** Homework (20%), Mid-term (30%), and Final project (50%)

• **Course Webpage:** [http://people.oregonstate.edu/~fuxia/ece599.html](http://people.oregonstate.edu/~fuxia/ece599.html)

• **Instructor:** Xiao Fu (Kelley 3003, xiao.fu@oregonstate.edu)

• **TA:** Ms. Sharmin Kibria (kibrias@oregonstate.edu)

• **Office Hours:** Thur 9:00-10:00 am & 3:00-4:00 pm

• **Lecture time:** Tue and Thur 12:00 pm - 1:50 pm
Course Objectives

- **Course Objectives**: after this term, I expect you to
  - understand the basic concepts of convex optimization, e.g., convex function, convex set, optimality conditions;
  - master the skill of formulating engineering problems as (hopefully convex) optimization problems;
  - be able to design efficient algorithms for your own problems in machine learning, data mining, signal processing, etc.
Tentative Course Outline

• Topics to cover:
  – Introduction
  – Review of Linear Algebra
  – Convex Set
  – Convex Function
  – Convex Problems
  – Duality
  – Gradient Descent, Newton Method, and Interior Point Method
  – Accelerated Gradient and Proximal/Projected Gradient
  – Stochastic Gradient Descent (SGD) - \textbf{How to train deep neural nets?}
  – Alternating Direction Method of Multipliers (ADMM)
  – Nonconvex Optimization
Textbook

- Textbook:
  
Optimization

- A generic description of optimization:

\[
\min_x f_0(x) \\
\text{s.t. } f_i(x) \leq b_i, \ i = 1, \ldots, m,
\]

where

- \( x = (x_1, \ldots, x_n) = [x_1, \ldots, x_n]^T \in \mathbb{R}^n \) is the optimization variable
- \( f_0(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is the objective function
- \( f_i(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) for \( i = 1, \ldots, m \), and \( f_i(x) \leq b_i \) are the constraints.

- All \( z \)'s that satisfy \( f_i(z) \leq b_i \) for \( i = 1, \ldots, m \) are called feasible solutions.

- \( x^* \) is an optimal solution if \( f_0(x^*) \) gives the smallest value of \( f_0(z) \) among all feasible solutions; i.e. \( f(x^*) \leq f(z) \) for all feasible \( z \).

  - \( x^* \) is a feasible solution; \( f_0(x^*) \) is called the optimal value.
Why Should I Care?

- Optimization has become one of the most basic tools in EE&CS research.

- It is a branch of mathematics, but greatly helps formulate and solve engineering problems.

- It bridges EE and CS and provides a **unified view** for different applications:
  - Transmit beamforming (EE)
  - Resource allocation (EE)
  - Image denoising (EE)
  - Audio separation (EE)
  - Neural network training (CS)
  - Support vector machines (CS)
  - Clustering (CS)
  - ...

ECE599/CS519 Convex Optimization

X. Fu, School of EECS, Oregon State University
Diet Problem
Diet Problem

- **Problem**: find the cheapest diet such that the minimum nutrient requirements are fulfilled.
  
  - $x_j$ is the quantity of food $j$ for $j = 1, \ldots, n$.
  - Each unit of food $j$ has a cost of $c_j$.
  - One unit of food $j$ contains an amount $a_{ij}$ of nutrient $i$ for $i = 1, \ldots, m$.
  - We want nutrient $i$ to be at least equal to $b_i$ for $i = 1, \ldots, m$.

- **Objective**: cheapest diet $\sum_{j=1}^{n} c_j x_j$.

- **Optimization variables**: $x_j$, quantity of food $j$.

- **Constraints**: the smallest amount of nutrient $i$ ($b_i$) must be satisfied; i.e.

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, 2, \ldots, m$$
Diet Problem

- This problem can be formulated as:

\[
\begin{align*}
\min_x & \quad \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, 2, \ldots, m \\
& \quad x_j \geq 0, \quad j = 1, 2, \ldots, n
\end{align*}
\]

This is a **linear program**.

- Note that \(x_i \geq 0\) is by physical meaning.
Linear Prediction (LP)

- Let $y[0], y[1], \cdots$ be a time series.

- **Model** (autoregressive (AR) model):

  $$y[n] = a_1 y[n-1] + a_2 y[n-2] + \cdots + a_L y[n-L] + w[n], \quad n = 0, 1, 2, \ldots$$

  for some coefficients $\{a_i\}_{i=1}^L$, where $w[n]$ is noise or modeling error.

- **Problem:** estimate $\{a_i\}_{i=1}^L$ from $\{y[n]\}_{n=0}^M$.

- **Prediction:** $\hat{y}[n+1] \approx \sum_{i=0}^{L-1} \hat{a}_i y[n-L]$. 
Linear Regression

- We wish to find $a_1, \ldots, a_L$ such that $y[n]$ matches with $a_1 y[n - 1] + a_2 y[n - 2] + \cdots + a_L y[n - L]$ for every $n$.

- The optimization problem can be

$$
\min_{a} \sum_{n=L}^{M} (y[n] - (a_1 y[n - 1] + a_2 y[n - 2] + \cdots + a_L y[n - L]))^2.
$$

where $a = [a_1, \ldots, a_L]^T$.

- This is a **quadratic program**.

- **Applications:** time-series prediction, speech analysis and coding, spectral estimation...
blue— Hang Seng Index during a certain time period.
red— training phase; the line is $\sum_{l=1}^{L} a_l y[n - l]$; $a$ is obtained by LS; $L = 10$.
green— prediction phase; the line is $\hat{y}[n] = \sum_{l=1}^{L} a_l \hat{y}[n - l]$; the same $a$ in the training phase.
Classification

- Classification is one of the most basic tasks in machine learning.
Classification via Linear Classifier

training data

training data
Classification via Linear Classifier
Classification via Linear Classifier

- **Problem:** Given training pairs \( \{y_i, x_i\}_{i=1}^m \) where \( y_i \in \{-1, 1\} \) denotes the label and \( x_i \in \mathbb{R}^n \) is a feature vector of the \( i \)th training sample, respectively, classification aims at training a “classifier” that can determine the label of a new data when seeing the features \( x_{\text{new}} \).

- Training a linear classifier amounts to finding a ‘line’ in space, i.e., \( a^T x + b = 0 \), such that \( a^T x_i + b \geq 0 \) when \( x_i \in \text{class 1} (y_i = +1) \) and \( a^T x_i + b \leq 0 \) when \( x_i \in \text{class 2} (y_i = -1) \).

- The training problem can be formulated as

\[
\begin{align*}
\text{find} & \quad a, b \\
\text{s.t.} & \quad y_i(a^T x_i + b) \geq 0, \ \forall i.
\end{align*}
\]

This is a **feasibility problem** that can be converted to a linear program.

- **Testing:** \( y_{\text{new}} = \text{sign}(a^T x_{\text{new}} + b) \).


Clustering

- Similar to classification, but without training.

- It is natural for human to judge which figures should be clustered together - but how to design an algorithm to do the same?
Clustering

- The data vectors that are within the same cluster are similar to each other.
- One point in each cluster may be enough to represent others.
Clustering

- For example, the ‘centroids’ of the clusters can be good representatives.
Clustering: K-means

- **Problem description**: Given a set of data vectors \( x_1, \ldots, x_m \in \mathbb{R}^n \), find \( k \) centroids \( M = [M_1, \ldots, M_k] \in \mathbb{R}^{n \times k} \) of \( k \) clusters. In addition, assign each data vector to its closest centroid.

- **Problem formulation**

\[
\begin{align*}
\min_{M,\{s_j\}_{j=1}^m} & \quad \sum_{j=1}^m \|x_j - Ms_j\|^2_2 \\
\text{s.t.} & \quad \|s_j\|_0 = 1, \ s_{i,j} \in \{0, 1\},
\end{align*}
\]

where \( s_j \in \mathbb{R}^k \) is the ‘membership vector’ of data \( x_j \); i.e., \( s_{i,j} = 1 \) means \( x_j \) belongs to cluster \( i \).

- Note that \( s_j \) has only one nonzero element.

- Nonconvex program; but can be relaxed to a **semidefinite program**.
Direction-of-Arrival (DOA) estimation

\[ S_1,t \quad S_2,t \quad \ldots \quad S_{k-1},t \quad S_k,t \]

\[ Y_1,t \quad Y_2,t \quad \ldots \quad Y_{m-1},t \quad Y_m,t \]

\[ \theta_1 \quad \theta_2 \quad \theta_{k-1} \quad \theta_k \]

\[ d \]

ECE599/CS519 Convex Optimization

X. Fu, School of EECS, Oregon State University
Direction-of-Arrival (DOA) estimation

• Considering $t = 1, \ldots, p$, the signal model is

$$Y = A(\theta)S + N,$$

where

$$A(\theta) = \begin{bmatrix}
1 & \ldots & 1 \\
\frac{e^{-j2\pi d\sin(\theta_1)}}{\gamma} & \ldots & \frac{e^{-j2\pi d\sin(\theta_m)}}{\gamma} \\
\vdots & \ddots & \vdots \\
\frac{e^{-j2\pi d(n-1)\sin(\theta_1)}}{\gamma} & \ldots & \frac{e^{-j2\pi d(n-1)\sin(\theta_m)}}{\gamma}
\end{bmatrix}$$

$Y \in \mathbb{R}^{m \times p}$ are received signals, $S \in \mathbb{R}^{k \times p}$ sources, $N \in \mathbb{R}^{m \times p}$ noise, $m$ and $k$ number of receivers and sources, and $\gamma$ is the wavelength.

• **Objective:** estimate $\theta = [\theta_1, \ldots, \theta_k]^T$, where $\theta_i \in [-90^\circ, 90^\circ]$ for $i = 1, \ldots, k$. 
Direction-of-Arrival (DOA) estimation

• Let \( a(\theta) = \begin{bmatrix} 1, e^{-\frac{j2\pi d \sin(\theta)}{\gamma}}, \ldots, e^{-\frac{j2(n-1)\pi d \sin(\theta)}} \end{bmatrix}^T. \)

• Consider constructing a steering response library, say, by \(1^\circ\) uniform sampling

\[
A = \begin{bmatrix} a(-90^\circ), a(-89^\circ), a(-88^\circ), \ldots, a(88^\circ), a(89^\circ), a(90^\circ) \end{bmatrix},
\]

(Obviously you can also make the sampling more dense, or perhaps non-uniform).

• The true steering response matrix \( A(\theta) \) is, in an approximate sense, a submatrix of \( A \).

• DOA estimation is approximately equivalent to finding the columns of \( A(\theta) \) in \( A \).
Direction-of-Arrival (DOA) estimation

- Example: \( k = 3, \theta = [-90^\circ, -88^\circ, 88^\circ] \).

\[
\begin{align*}
Y &= A \cdot X \\
&= \begin{bmatrix}
S_{1,1}, & \ldots, & S_{1,p} \\
S_{2,1}, & \ldots, & S_{2,p} \\
\vdots \\
S_{3,1}, & \ldots, & S_{3,p}
\end{bmatrix}
\end{align*}
\]

- To locate the “active columns” in \( A \) is equivalent to find a row-sparse \( X \).

- Problem formulation:

\[
\min_X \|Y - AX\|_F^2 + \lambda \|X\|_{2,1}.
\]
Direction-of-Arrival (DOA) estimation

- This is a group-lasso problem in sparse optimization – nonsmooth convex optimization which can be recast as a second-order cone program.

- **Simulation:** $k = 3$, $p = 100$, $n = 8$ and $\text{SNR} = 30$ dB; three sources come from $-65^\circ$, $-20^\circ$ and $42^\circ$, respectively. $A = [a(-90^\circ), a(-89.5^\circ), \ldots, a(90^\circ)] \in \mathbb{R}^{m \times 381}$.

![Graph 1](image1.png)  
![Graph 2](image2.png)

True Angle $\theta = [-65, -20, 42]^T$  
Values of $|X|$
Video Analysis

- Suppose that we are given video sequences $F_i, i = 1, \ldots, p$.

- Our objective is to extract the background in the video sequences.

- The background is of low-rank, as the background is static within a short period of time.

- The foreground is sparse, as activities in the foreground only occupy a small fraction of space.
Video Analysis

- Stacking the video sequences \( Y = [\text{vec}(F_1), \ldots, \text{vec}(F_p)] \), we have

\[
Y = X + E,
\]

where \( X \) represents the low-rank background, and \( E \) the sparse foreground.

- Nuclear norm and \( \ell_1 \)-norm based formulation:

\[
\min_{X, E} \|X\|_* + \gamma \|\text{vec}(E)\|_1
\]

\[
\text{s.t. } Y = X + E.
\]

- Nuclear norm promotes low-rank and \( \ell_1 \) norm promotes sparsity.

- This is still a convex optimization problem.
Video Analysis

- 500 images, image size $160 \times 128$, $\gamma = 1/\sqrt{160 \times 128}$.
  - Row 1: the original video sequences.
  - Row 2: the extracted low-rank background.
  - Row 3: the extracted sparse foreground.
Recommender System

- Now you go to Netflix, you always see this:
- Design algorithms to defeat Netflix's recommender system and win 1 million dollars.
Matrix Completion

• **Problem**: Consider an incomplete user-by-movie matrix $X$, where $X_{i,j}$ denotes the recorded score of movie $j$ given by user $i$. Only a portion of $(i,j)$ is observed. Can we predict the unobserved $X_{i,j}$’s accurately?

• **Intuition**: The complete $X$ matrix is low-rank, since the user preferences are not random. In other words, users can be roughly clustered into several groups and movies can also be clustered into different categories.

• **Problem formulation**:

$$\min_{M} \|M\|_*$$

s.t. $M_{i,j} = X_{i,j}, \forall (i,j) \in \Omega,$

where $\Omega$ denotes the observed set.
Optimization is Nontrivial

- Many engineering problems boil down to optimization problems.

- **Most** optimization problems are very hard.
  - Being hard means no polynomial-time solvers is known.

- The class of polynomial-time solvable problems is just a small portion of optimization problems.

- History:
  - Before the 1980s: Linear Programming v.s. Nonlinear Programming
  - After the 1980s: Convex Optimization v.s. Nonconvex Optimization

- **Interior-point method**:
  - proposed by Narendra Karmarkar to solve linear programs in 1984.
  - generalized by Yurii Nesterov and Arkadi Nemirovskii to cover convex opt.
Nonconvex Problems

- Many more real-life problems are nonconvex.

- The role of convex optimization in handling nonconvex problems
  - block convex optimization (alternating optimization)
  - convex relaxation/approximation
  - successive convex approximation

In a nutshell, many nonconvex problems can be handled by solving one or a series of convex subproblems.
Nonconvex Problems

• What's the difference between convex and nonconvex functions

convex
strictly convex
non-convex

• Note: for most nonconvex problems, local minima is the best we can hope for.
Nonconvex Optimization: Topic Mining

Topic 1: Clinton, White House, Scandal, Lewinsky, grand jury...

Topic 2: Utah, Chicago, NBA, Jordan, Carl, jazz, bull, basketball, final...

Topic 3: NASA, Columbia, shuttle, space, experiments, ...
Topic Mining

\[ x_j \approx 0.2 \times a_1 + 0.65 \times a_2 + 0.15 \times a_3 \]

\[ \text{doc. } \approx 0.2 \times \text{economy} + 0.65 \times \text{politics} + 0.15 \times \text{history} \]
Topic Mining: Formulation

- Problem formulation

\[
\min_{\{a_i\}_{j=1}^k, \{s_j\}_j} \sum_{j=1}^m \left\| x_j - \sum_{i=1}^k a_i s_{i,j} \right\|_2^2
\]

s.t. \( s_j \geq 0, \ a_i \geq 0 \).

- called nonnegative matrix factorization; nonconvex optimization.
Topic Mining: Algorithm

• Solving NMF via convex subproblems:

\[
\text{step 1} \quad \{a_i^+\} \leftarrow \arg\min_{\{a_i\}} \sum_{j=1}^m \left\| x_j - \sum_{i=1}^k a_i s_{i,j} \right\|^2 \quad \text{s.t. } a_i \geq 0
\]

\[
\text{step 2} \quad \{s_j^+\} \leftarrow \arg\min_{\{s_j\}} \sum_{j=1}^m \left\| x_j - \sum_{i=1}^k a_i^+ s_{i,j} \right\|^2 \quad \text{s.t. } s_j \geq 0
\]

\[
\text{step 3} \quad a_i \leftarrow a_i^+, \quad s_j \leftarrow s_j^+
\]

\text{goto step 1}

• Steps 1 and 2 are both convex optimization problems.

• Many NMF algorithms are based on the above idea.
Some Recent Results

Table 1: Mined topics from 5 classes of (1,683) articles of the TDT2 corpus.

<table>
<thead>
<tr>
<th>Allegations</th>
<th>FastAnchor</th>
<th>AnchorFree (proposed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>allegations</td>
<td>Shutter</td>
<td>Shutter</td>
</tr>
<tr>
<td>lewinsky</td>
<td>Columbia</td>
<td>Columbia</td>
</tr>
<tr>
<td>clinton</td>
<td>Motors</td>
<td>Motors</td>
</tr>
<tr>
<td>lady</td>
<td>Workers</td>
<td>Workers</td>
</tr>
<tr>
<td>white</td>
<td>Utah</td>
<td>Utah</td>
</tr>
<tr>
<td>hillary</td>
<td>Final</td>
<td>Final</td>
</tr>
<tr>
<td>monica</td>
<td>Gm</td>
<td>Gm</td>
</tr>
<tr>
<td>starr</td>
<td>Jazz</td>
<td>Jazz</td>
</tr>
<tr>
<td>house</td>
<td>Nba</td>
<td>Nba</td>
</tr>
<tr>
<td>husband</td>
<td>Utah</td>
<td>Utah</td>
</tr>
<tr>
<td>husband</td>
<td>Workers</td>
<td>Workers</td>
</tr>
<tr>
<td>husband</td>
<td>Michigan</td>
<td>Michigan</td>
</tr>
<tr>
<td>husband</td>
<td>Chicago</td>
<td>Chicago</td>
</tr>
<tr>
<td>husband</td>
<td>Jordan</td>
<td>Jordan</td>
</tr>
<tr>
<td>house</td>
<td>House</td>
<td>House</td>
</tr>
<tr>
<td>house</td>
<td>Mississippi</td>
<td>Mississippi</td>
</tr>
<tr>
<td>house</td>
<td>Auto</td>
<td>Auto</td>
</tr>
<tr>
<td>house</td>
<td>Plant</td>
<td>Plant</td>
</tr>
<tr>
<td>house</td>
<td>Strike</td>
<td>Strike</td>
</tr>
<tr>
<td>house</td>
<td>Gms</td>
<td>Gms</td>
</tr>
<tr>
<td>house</td>
<td>Michael</td>
<td>Michael</td>
</tr>
<tr>
<td>house</td>
<td>Championship</td>
<td>Championship</td>
</tr>
<tr>
<td>house</td>
<td>Tonight</td>
<td>Tonight</td>
</tr>
<tr>
<td>house</td>
<td>Win</td>
<td>Win</td>
</tr>
<tr>
<td>house</td>
<td>Karl</td>
<td>Karl</td>
</tr>
<tr>
<td>starr</td>
<td>Lewinsky</td>
<td>Lewinsky</td>
</tr>
<tr>
<td>starr</td>
<td>Games</td>
<td>Games</td>
</tr>
<tr>
<td>starr</td>
<td>Basketball</td>
<td>Basketball</td>
</tr>
<tr>
<td>starr</td>
<td>Night</td>
<td>Night</td>
</tr>
<tr>
<td>starr</td>
<td>Stars</td>
<td>Stars</td>
</tr>
<tr>
<td>starr</td>
<td>Gms</td>
<td>Gms</td>
</tr>
<tr>
<td>starr</td>
<td>Autoworkers</td>
<td>Autoworkers</td>
</tr>
<tr>
<td>starr</td>
<td>Walkouts</td>
<td>Walkouts</td>
</tr>
<tr>
<td>starr</td>
<td>Shuttle</td>
<td>Shuttle</td>
</tr>
<tr>
<td>starr</td>
<td>Space</td>
<td>Space</td>
</tr>
<tr>
<td>starr</td>
<td>Astronauts</td>
<td>Astronauts</td>
</tr>
<tr>
<td>starr</td>
<td>Nasa</td>
<td>Nasa</td>
</tr>
<tr>
<td>starr</td>
<td>Crew</td>
<td>Crew</td>
</tr>
<tr>
<td>starr</td>
<td>Utah</td>
<td>Utah</td>
</tr>
<tr>
<td>starr</td>
<td>Finals</td>
<td>Finals</td>
</tr>
<tr>
<td>starr</td>
<td>Experiments</td>
<td>Experiments</td>
</tr>
<tr>
<td>starr</td>
<td>Gms</td>
<td>Gms</td>
</tr>
<tr>
<td>starr</td>
<td>Michael</td>
<td>Michael</td>
</tr>
<tr>
<td>starr</td>
<td>Jay</td>
<td>Jay</td>
</tr>
<tr>
<td>starr</td>
<td>Tigers</td>
<td>Tigers</td>
</tr>
<tr>
<td>starr</td>
<td>Union</td>
<td>Union</td>
</tr>
<tr>
<td>starr</td>
<td>Idled</td>
<td>Idled</td>
</tr>
<tr>
<td>starr</td>
<td>System</td>
<td>System</td>
</tr>
<tr>
<td>starr</td>
<td>Production</td>
<td>Production</td>
</tr>
<tr>
<td>starr</td>
<td>Walkouts</td>
<td>Walkouts</td>
</tr>
<tr>
<td>starr</td>
<td>Weightlessness</td>
<td>Weightlessness</td>
</tr>
<tr>
<td>starr</td>
<td>Earth</td>
<td>Earth</td>
</tr>
<tr>
<td>starr</td>
<td>Mice</td>
<td>Mice</td>
</tr>
<tr>
<td>starr</td>
<td>Animals</td>
<td>Animals</td>
</tr>
<tr>
<td>starr</td>
<td>Fish</td>
<td>Fish</td>
</tr>
<tr>
<td>starr</td>
<td>Neurological</td>
<td>Neurological</td>
</tr>
<tr>
<td>starr</td>
<td>Seven</td>
<td>Seven</td>
</tr>
<tr>
<td>starr</td>
<td>Title</td>
<td>Title</td>
</tr>
<tr>
<td>starr</td>
<td>13year</td>
<td>13year</td>
</tr>
<tr>
<td>starr</td>
<td>Girls</td>
<td>Girls</td>
</tr>
<tr>
<td>starr</td>
<td>Shootings</td>
<td>Shootings</td>
</tr>
<tr>
<td>starr</td>
<td>Karl</td>
<td>Karl</td>
</tr>
<tr>
<td>starr</td>
<td>Pippen</td>
<td>Pippen</td>
</tr>
<tr>
<td>starr</td>
<td>Basketball</td>
<td>Basketball</td>
</tr>
<tr>
<td>starr</td>
<td>Night</td>
<td>Night</td>
</tr>
<tr>
<td>starr</td>
<td>Sixth</td>
<td>Sixth</td>
</tr>
<tr>
<td>starr</td>
<td>Games</td>
<td>Games</td>
</tr>
<tr>
<td>starr</td>
<td>Title</td>
<td>Title</td>
</tr>
<tr>
<td>starr</td>
<td>Johnson</td>
<td>Johnson</td>
</tr>
</tbody>
</table>

ECE599/CS519 Convex Optimization

X. Fu, School of EECS, Oregon State University
Sparse Recovery

- Consider a noiseless model
  \[ y = Ax \]
  but with \( m < n \) (underdetermined system).

- By linear algebra, there are many (infinite) possible solutions to \( y = Ax \)
  \[ Ax = y \iff x = A^\dagger y + u, \ u \in \mathcal{N}(A) \]
  where \( A^\dagger = A^T (AA^T)^{-1} \), and \( \mathcal{N}(A) = \{ u \mid Au = 0 \} \) is the nullspace of \( A \).

- How can we choose \( x \) from these possible solutions?

- This is what matrix textbooks would tell us to do: Least 2-norm:
  \[
  \min \|x\|_2 \\
  \text{s.t. } Ax = y
  \]
  The solution has a closed form, \( x^* = A^\dagger y \) (make sense intuitively).
Sparse recovery

• Least 0-norm reconstruction:

\[
\begin{align*}
\min & \quad \|x\|_0 \\
\text{s.t.} & \quad Ax = y
\end{align*}
\]

where \(\|x\|_0\) counts the number of nonzero elements in \(x\).

• Make sense for sparse signals; i.e., signals with many zeros.

• Can prove that if the no. of zeros in the actual \(x\) is sufficiently large compared to the no. of measurements \(m\), then 0-norm minimization leads to the ground truth.

• \(\|\cdot\|_0\) is not convex. In fact, 0-norm minimization poses a very hard problem.
Convex Approximation

• Least 1-norm reconstruction:

\[
\min \|x\|_1 \\
\text{s.t. } Ax = y
\]

• the ‘best’ convex approximation to 0-norm.

• can prove that under some assumptions, 1-norm minimization is able to approach 0-norm minimization (in some probabilistic sense).

• Currently a very hot topic (in the literature it is called compressive sensing)

• 1-norm minimization is an LP

\[
\min \sum_{i=1}^{n} t_i \\
\text{s.t. } Ax = y \\
-t_i \leq x_i \leq t_i, \quad i = 1, \ldots, n
\]
Simulation

1–norm reconstruction

2–norm reconstruction
Some Remarks for Your Research

• Convex problems are very nice – provably solvable in polynomial time.

• But solving a particular convex optimization problem may be very expensive -
convexity does not mean every thing!

• Nonconvex problems are in general hard, but nonconvexity ≠ hardness.
  – There are many solvable nonconvex problems, e.g., the principal component
analysis problem.

• Many problems are nonconvex, but can be (mysteriously) solved to very good
solutions – so don’t be scared by nonconvexity.

• Bottom line: analyze your own problem case by case on top of convexity, and run
simulations and experiments!
Take-home Points

• Convex optimization is essential for engineering.

• We have seen many machine learning, data mining, and signal processing examples. There are many more.

• Nonconvex optimization can also be handled using convex optimization techniques.

• **Next lecture:** we'll review some mathematical background.