# Sparse Optimization
# Lecture: Basic Sparse Optimization Models

Instructor: Wotao Yin
Department of Mathematics, UCLA
July 2013

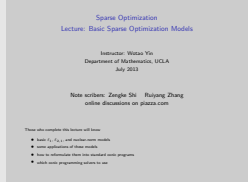
Note scribers: Zengke Shi  Ruiyang Zhang
online discussions on piazza.com

Those who complete this lecture will know

- basic $\ell_1$, $\ell_{2,1}$, and nuclear-norm models
- some applications of these models
- how to reformulate them into standard conic programs
- which conic programming solvers to use







## Examples of Sparse Optimization Applications

See online seminar at piazza.com



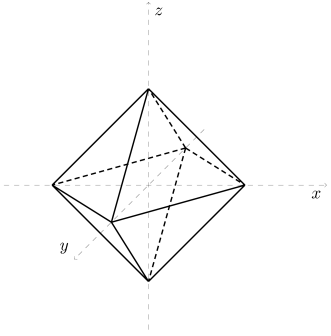
Examples of Sparse Optimization Applications

See online seminar at piazza.com

## Basis pursuit

$$\min\{\|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

- find least $\ell_1$-norm point on the affine plane $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$
- tends to return a sparse point (sometimes, the sparsest)



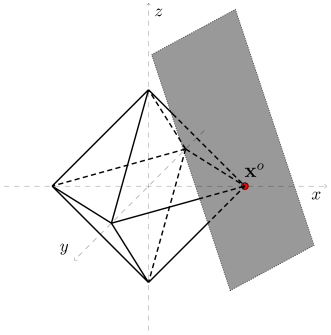

$\ell_1$ ball

- "$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{C}\}$" is a compact way of describing an optimization problem. The function before ":" is the objective function and everything behind ":" describes the feasible set or the constraints.
- The diamond in the figure is the isosurface of the $\ell_1$-norm.

# Basis pursuit

$$\min\{\|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

- find least $\ell_1$-norm point on the affine plane $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$
- tends to return a sparse point (sometimes, the sparsest)




$\ell_1$ ball touches the affine plane

- The optimization problem is geometrically interpreted in the figure. The diamond (isosurface of the $\ell_1$-norm) grows until it touches the plane ($\mathbf{A}\mathbf{x} = \mathbf{b}$).
- Why returns a sparse point?

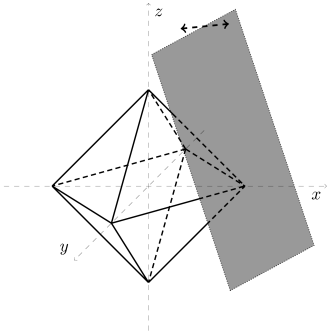
  As we can see in the figure, the borders of the diamond are pretty sharp. It is very likely that only one vertex, rather than an edge or a face, touches the plane first. So the solution is just a sparse point.
- However, it is possible that an entire edge or even an entire face of the diamond touch the plane simultaneously. Thus, the solution may not be unique.

# Basis pursuit denoising, LASSO

$$\min_{\mathbf{x}}\{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 : \|\mathbf{x}\|_1 \le \tau\}, \tag{1a}$$

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2, \tag{1b}$$

$$\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \le \sigma\}. \tag{1c}$$




all models allow $\mathbf{A}\mathbf{x}^* \neq \mathbf{b}$

- $\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \le \sigma$ allows the affine plane $\mathbf{A}\mathbf{x}=\mathbf{b}$ to move along its normal directions, i.e., represented by the row vectors of $\mathbf{A}$, for up to $\sigma$.

## Basis pursuit denoising, LASSO

$$\min_{\mathbf{x}}\{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 : \|\mathbf{x}\|_1 \leq \tau\}, \tag{2a}$$

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2, \tag{2b}$$

$$\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \leq \sigma\}. \tag{2c}$$

- $\|\cdot\|_2$ is most common for error but can be generalized to loss function $\mathcal{L}$
- (2a) seeks for a least-squares solution with "bounded sparsity"
- (2b) is known as LASSO (least absolute shrinkage and selection operator). it seeks for a balance between sparsity and fitting
- (2c) is referred to as BPDN (basis pursuit denoising), seeking for a sparse solution from tube-like set $\{\mathbf{x} : \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \leq \sigma\}$
- they are equivalent (see later slides)
- in terms of regression, they select a (sparse) set of features (i.e., columns of $\mathbf{A}$) to linearly express the observation $\mathbf{b}$





Basis pursuit denoising, LASSO

$\min_{\mathbf{x}}\{\|\mathbf{Ax}-\mathbf{b}\|_2 : \|\mathbf{x}\|_1 \leq \tau\}$. (2a)

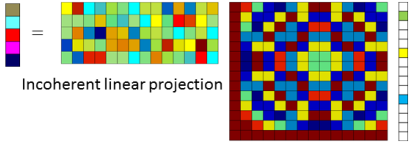
$\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2$. (2b)

$\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \|\mathbf{Ax}-\mathbf{b}\|_2 \leq \sigma\}$. (2c)

- $\|\cdot\|_2$ is most common for error but can be generalized to loss function $\mathcal{L}$
- (2a) seeks for a least-squares solution with "bounded sparsity"
- (2b) is known as LASSO (least absolute shrinkage and selection operator). it seeks for a balance between sparsity and fitting
- (2c) is referred to as BPDN (basis pursuit denoising), seeking for a sparse solution from tube-like set $\{\mathbf{x} : \|\mathbf{Ax}-\mathbf{b}\|_2 \leq \sigma\}$
- they are equivalent (see later slides)
- in terms of regression, they select a (sparse) set of features (i.e., columns of $\mathbf{A}$) to linearly express the observation $\mathbf{b}$

## Sparse under basis $\Psi$ / $\ell_1$-synthesis model

$$\min_{\mathbf{s}}\{\|\mathbf{s}\|_1 : \mathbf{A}\Psi\mathbf{s} = \mathbf{b}\} \tag{3}$$




- signal $\mathbf{x}$ is *sparsely synthesized* by atoms from $\Psi$, so vector $\mathbf{s}$ is sparse
- $\Psi$ is referred to as the *dictionary*
- commonly used dictionaries include both analytic and trained ones
- analytic examples: Id, DCT, wavelets, curvelets, gabor, etc., also their combinations; they have analytic properties, often easy to compute (for example, multiplying a vector takes $O(n \log n)$ instead of $O(n^2)$)
- $\Psi$ can also be numerically learned from *training data* or *partial signal*
- they can be orthogonal, frame, or general

- $\mathbf{\Psi s} = \mathbf{x}$. In the example of compressive imaging, $\mathbf{x}$ is the original image, $\mathbf{A}$ is the sensing matrix, and $\mathbf{Ax}$ is implemented by the sensing device. $\mathbf{b}$ is the recorded measurements. By using the *dictionary* $\mathbf{\Psi}$, we can get a sparse vector $\mathbf{s}$ instead of $\mathbf{x}$, which makes it convenient to store or transmit the image.
- Different *dictionaries* are used/created according to different features in the images. Even the size of the *dictionary* $\mathbf{\Psi}$ is subject to determination.
- DCT: discrete cosine transform.

# Sparse under basis $\Psi$ / $\ell_1$-synthesis model

If $\Psi$ is **orthogonal**, problem (3) is equivalent to

$$\min_{\mathbf{x}}\{\|\Psi^*\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{4}$$

by change of variable $\mathbf{x} = \Psi\mathbf{s}$, equivalently $\mathbf{s} = \Psi^*\mathbf{x}$.

Related models for noise and approximate sparsity:

$$\min_{\mathbf{x}}\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 : \|\Psi^*\mathbf{x}\|_1 \le \tau\},$$
$$\min_{\mathbf{x}} \|\Psi^*\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2,$$
$$\min_{\mathbf{x}}\{\|\Psi^*\mathbf{x}\|_1 : \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \le \sigma\}.$$



- This model is more difficult to solve and to analyze than the one on the last page, because the objective function is no longer seperable.

# Sparse after transform / $\ell_1$-analysis model

$$\min_{\mathbf{x}}\{\|\Psi^*\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{5}$$

Signal $\mathbf{x}$ becomes sparse under the transform $\Psi$ (may not be orthogonal)

**Examples** of $\Psi$:

- DCT, wavelets, curvelets, ridgelets, ....
- tight frames, Gabor, ...
- (weighted) total variation

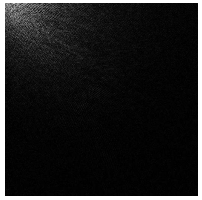
When $\Psi$ is not orthogonal, the analysis is more difficult

# Example: sparsify an image



Example: sparsify an image

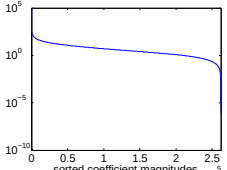
This is Cameraman, one of the most frequently used test images.



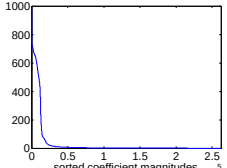
(a) DCT coefficients

(b) Haar wavelets

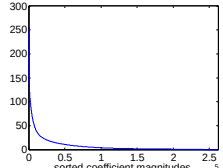
(c) Local variation




(d) DCT coeff's decay




(e) Haar wavelets




(f) Local variation

Figure: the DCT and wavelet coefficients are scaled for better visibility.

- Figure (a), (b), and (c) are three different kinds of transforms applied to Cameraman.
- Figure (c) represents the sizes of all local gradients for the original image. Given all the local gradients, to restore the original image, we also need the average value of all the pixels.
- The idea of compressing the image is to keep the largest coefficients and discard others. In the figure (e), the curve for the magnitudes of sorted coefficients generated by Harr wavelets method is pretty steep, and thus the compression would incur very little loss of information and could almost restore the original image.

# Questions

1. Can we trust these models to return intended sparse solutions?
2. When will the solution be unique?
3. Will the solution be robust to noise in $\mathbf{b}$?
4. Are constrained and unconstrained models equivalent? in what sense?

   Questions 1–4 will be addressed in *next lecture*.

5. How to choose parameters?
   - $\tau$ (sparsity), $\mu$ (weight), and $\sigma$ (noise level) have different meanings
   - applications determine which one is easier to set
   - generality: use a test data set, then scale parameters for real data
   - cross validation: reserve a subset of data to test the solution



Questions

1. Can we trust these models to return intended sparse solutions?
2. When will the solution be unique?
3. Will the solution be robust to noise in b?
4. Are constrained and unconstrained models equivalent? in what sense?

Questions 1–4 will be addressed in *next lecture*.

5. How to choose parameters?
   - τ (sparsity), μ (weight), and σ (noise level) have different meanings
   - applications determine which one is easier to set
   - generality: use a test data set, then scale parameters for real data
   - cross validation: reserve a subset of data to test the solution

More remarks on choosing parameters:

- In the unconstrained model, we could see that $\mathbf{x}$ is of different orders in $\|\cdot\|_1$ and $\|\cdot\|_2^2$. Therefore, we should adjust the weight $\mu$ properly when the data and solutions have different sizes and scales.
- *Cross validation* is a technique for assessing how the results of a model (and its parameter selection) will generalize to an independent data set. One can partition $(\mathbf{A}, \mathbf{b})$ to the training and testing datasets, run the model on the training set and validating the results (and the parameter selection) on the testing set.

## Joint/group sparsity

Joint sparse recovery model:

$$\min_{\mathbf{X}}\{\|\mathbf{X}\|_{2,1} : \mathcal{A}(\mathbf{X}) = \mathbf{b}\} \tag{6}$$

where

$$\|\mathbf{X}\|_{2,1} := \sum_{i=1}^{m} \|[x_{i1}\ x_{i,2} \cdots x_{in}]\|_2 .$$

- $\ell_2$-norm is applied to each row of $\mathbf{X}$
- $\ell_{2,1}$-norm ball has sharp boundaries "across different rows", which tend to be touched by $\{\mathbf{X} : \mathcal{A}(\mathbf{X}) = \mathbf{b}\}$, so the solution tends to be *row-sparse*
- also $\|\mathbf{X}\|_{p,q}$ for $1 < p \leq \infty$, affects magnitudes of entries on the same row
- complex-valued signals are a special case





Joint/group sparsity

Joint sparse recovery model:

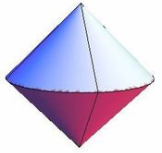
$$\min_{\mathbf{X}}\{\|\mathbf{X}\|_{2,1} : \mathcal{A}(\mathbf{X}) = \mathbf{b}\} \tag{6}$$

where

$$\|\mathbf{X}\|_{2,1} := \sum_{i=1}^{m} \|[x_{i1}\ x_{i,2} \cdots x_{in}]\|_2 .$$

- $\ell_2$-norm is applied to each row of $\mathbf{X}$
- $\ell_{2,1}$-norm ball has sharp boundaries "across different rows", which tend to be touched by $\{\mathbf{X} : \mathcal{A}(\mathbf{X}) = \mathbf{b}\}$, so the solution tends to be *row-sparse*
- also $\|\mathbf{X}\|_{p,q}$ for $1 < p \leq \infty$, affects magnitudes of entries on the same row
- complex-valued signals are a special case

- Example of joint sparsity: $\|\mathbf{x}\| := |x_1| + \|(x_2, x_3)\|_2$



Isosurface of this norm

The two end points and horizontal circle of this figure are sharp so they tend to touch the plane $\mathbf{Ax} = \mathbf{b}$. They correspond to $x_1 \neq 0$ and $(x_2, x_3) \neq 0$, respectively. It is unlikely that all components of $\mathbf{x}$ are nonzero the same time.

- The norm $\|\cdot\|_{2,1}$ is the $\ell_1$-norm of row energies. Minimizing this objective function, we often obtain a solution $\mathbf{X}$ with only few non-zero rows (group sparsity).
- Norm for a complex vector is defined as the sum of the components' magnitudes.

# Joint/group sparsity

Decompose $\{1, \ldots, n\} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \cdots \cup \mathcal{G}_S$.

- non-overlapping groups: $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \ \forall i \neq j$.
- otherwise, groups may overlap (modeling many interesting structures).

Group-sparse recovery model:

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_{\mathcal{G},2,1} : \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{7}$$

where

$$\|\mathbf{x}\|_{\mathcal{G},2,1} = \sum_{s=1}^{S} w_s \|\mathbf{x}_{\mathcal{G}_s}\|_2.$$



Joint/group sparsity

Decompose $\{1, \ldots, n\} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \cdots \cup \mathcal{G}_S$.

- non-overlapping groups: $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset, \ \forall i \neq j$.
- otherwise, groups may overlap (modeling many interesting structures).

Group-sparse recovery model:

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_{\mathcal{G},2,1} : \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{7}$$

where

$$\|\mathbf{x}\|_{\mathcal{G},2,1} = \sum_{s=1}^{S} w_s \|\mathbf{x}_{\mathcal{G}_s}\|_2.$$

## Auxiliary constraints

Auxiliary constraints introduce additional structures of the underlying signal into its recovery, which sometimes *significantly* improve recovery quality

- nonnegativity: $\mathbf{x} \geq \mathbf{0}$
- bound (box) constraints: $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$
- general inequalities: $\mathbf{Qx} \leq \mathbf{q}$

They can be very effective in practice. They also generate "corners."



Auxiliary constraints
Auxiliary constraints introduce additional structures of the underlying signal into its recovery, which sometimes significantly improve recovery quality
• nonnegativity: $\mathbf{x} \geq \mathbf{0}$
• bound (box) constraints: $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$
• general inequalities: $\mathbf{Qx} \leq \mathbf{q}$
They can be very effective in practice. They also generate "corners."

## Reduce to conic programs

Sparse optimization often has *nonsmooth* objectives.

Classic conic programming solvers do not handle nonsmooth functions.

Basic idea: *model nonsmoothness by inequality constraints*.

Example: for given $\mathbf{x}$, we have

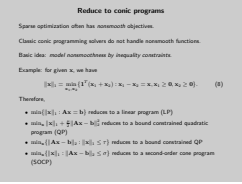
$$\|\mathbf{x}\|_1 = \min_{\mathbf{x}_1,\mathbf{x}_2} \{\mathbf{1}^T(\mathbf{x}_1 + \mathbf{x}_2) : \mathbf{x}_1 - \mathbf{x}_2 = \mathbf{x}, \mathbf{x}_1 \geq \mathbf{0}, \mathbf{x}_2 \geq \mathbf{0}\}. \qquad (8)$$

Therefore,

- $\min\{\|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ reduces to a linear program (LP)
- $\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ reduces to a bound constrained quadratic program (QP)
- $\min_{\mathbf{x}}\{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 : \|\mathbf{x}\|_1 \leq \tau\}$ reduces to a bound constrained QP
- $\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \sigma\}$ reduces to a second-order cone program (SOCP)








- In (8), we intend to decompose $\mathbf{x}$ into its *positive part* $\mathbf{x}_1 \geq 0$ and *negative part* $\mathbf{x}_2 \geq 0$. For example, if $\mathbf{x} = (5, -2)^T$, we want to have $\mathbf{x}_1 = (5, 0)^T$, $\mathbf{x}_2 = (0, 2)^T$. For every $i$, either $x_{1,i} =$ or $x_{2,i} = 0$ due to the minimization of $\mathbf{x}_1 + \mathbf{x}_2$.
- In QP (quadratic programming), quadratic terms are only allowed in the objective function. All constraints should be linear, equality or inequality, if there is any.

# Conic programming

Basic form:

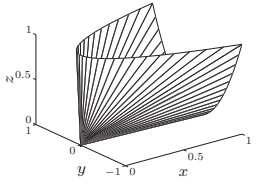
$$\min_{\mathbf{x}}\{\mathbf{c}^T\mathbf{x} : \mathbf{F}\mathbf{x}+\mathbf{g} \succeq_{\mathcal{K}} \mathbf{0}, \mathbf{A}\mathbf{x}=\mathbf{b}.\}$$

"$\mathbf{a} \succeq_{\mathcal{K}} \mathbf{b}$" stands for $\mathbf{a}-\mathbf{b} \in \mathcal{K}$, which is a convex, closed, pointed cone.

Examples:

- first orthant (cone): $\mathbb{R}^n_+ = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0}\}$.
- norm cone (2nd order cone): $\mathcal{Q} = \{(\mathbf{x}, t) : \|\mathbf{x}\| \leq t\}$
- polyhedral cone: $\mathcal{P} = \{\mathbf{x} : \mathbf{A}\mathbf{x} \geq \mathbf{0}\}$
- positive semidefinite cone: $\mathbf{S}_+ = \{\mathbf{X} : \mathbf{X} \succeq \mathbf{0}, \mathbf{X}^T = \mathbf{X}\}$
  **Example**:

$$\left\{(x,y,z) : \begin{bmatrix} x & y \\ y & z \end{bmatrix} \in \mathbf{S}_+\right\}$$




- A convex, closed, pointed cone is also called a proper convex cone. Pointed means that the cone contains no line.
- $\begin{bmatrix} x & y \\ y & z \end{bmatrix} \in \mathbf{S}_+$ iff

$$\begin{cases} x \geq 0 \\ z \geq 0 \\ xz \geq y^2 \end{cases}$$

## Linear program

Model

$$\min\{\mathbf{c}^T\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \succeq_{\mathcal{K}} \mathbf{0}\}$$

where $\mathcal{K}$ is the nonnegative cone (first orthant).

$$\mathbf{x} \succeq_{\mathcal{K}} \mathbf{0} \Longleftrightarrow \mathbf{x} \geq \mathbf{0}.$$

Algorithms

- the Simplex method (move between vertices)
- interior-point methods (IPMs) (move inside the polyhedron)
- decomposition approaches (divide and conquer)

In primal IPM, $\mathbf{x} \geq 0$ is replaced by its logarithmic barrier:

$$\psi(\mathbf{y}) = \sum_i \log(y_i)$$

log-barrier formulation:

$$\min\{\mathbf{c}^T\mathbf{x} - (1/t)\sum_i \log(x_i) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

# Second-order cone program

Model

$$\min\{\mathbf{c}^T\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \succeq_{\mathcal{K}} \mathbf{0}\}$$

where $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$; each $\mathcal{K}_k$ is the second-order cone

$$\mathcal{K}_k = \left\{\mathbf{y} \in \mathbb{R}^{n_k} : y_{n_k} \geq \sqrt{y_1^2 + \cdots + y_{n_k-1}^2}\right\}.$$

IPM is the standard solver (though other options also exist)

Log-barrier of $\mathcal{K}_k$:

$$\psi(\mathbf{y}) = \log\left(y_{n_k}^2 - (y_1^2 + \cdots + y_{n_k-1})\right)$$

- The Simplex method move its points between vertices. There is no efficient generalization of it to non-polyhedral cones.
- Interior-point methods move its points inside the feasible set. They can take large and efficient steps, and are easy to generalize from polyhedron to other convex sets.
- Log-barrier formulation eliminates the inequality constraints. One can project the gradient of the objective to the plane, $\mathbf{A}\mathbf{x} = \mathbf{b}$, to find a descent direction.

# Semi-definite program

Model

$$\min\{\mathbf{C} \bullet \mathbf{X} : \mathcal{A}(\mathbf{X}) = \mathbf{b}, \mathbf{X} \succeq_{\mathcal{K}} \mathbf{0}\}$$

where $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_K$; each $\mathcal{K}_k = \mathbf{S}_+^{n_k}$.

IPM is the standard solver (though other options also exist)

Log-barrier of $\mathbf{S}_+^{n_k}$ (still a concave function):

$$\psi(\mathbf{Y}) = \log\det(\mathbf{Y}).$$

- There are a few equivalent ways to express the inner product of two matrices $\mathbf{C}$ and $\mathbf{X}$ of the same size: $\mathbf{C} \bullet \mathbf{X} = \mathrm{tr}(\mathbf{C}^T\mathbf{X}) = \sum_{i,j} c_{ij}x_{ij} = \langle \mathbf{C}, \mathbf{X} \rangle$.
- Proof of the concavity of $\psi(\mathbf{Y})$ can be found on page 74, Boyd & Vandenberghe, *Convex Optimization*.

(from Boyd & Vandenberghe, *Convex Optimization*)

**properties** (without proof): for $y \succ_K 0$,

$$\nabla\psi(y) \succeq_{K^*} 0, \qquad y^T\nabla\psi(y) = \theta$$

- nonnegative orthant $\mathbf{R}^n_+$: $\psi(y) = \sum_{i=1}^n \log y_i$

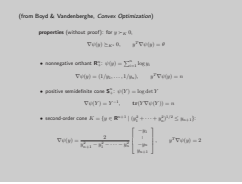
$$\nabla\psi(y) = (1/y_1, \ldots, 1/y_n), \qquad y^T\nabla\psi(y) = n$$

- positive semidefinite cone $\mathbf{S}^n_+$: $\psi(Y) = \log\det Y$

$$\nabla\psi(Y) = Y^{-1}, \qquad \mathbf{tr}(Y\nabla\psi(Y)) = n$$

- second-order cone $K = \{y \in \mathbf{R}^{n+1} \mid (y_1^2 + \cdots + y_n^2)^{1/2} \le y_{n+1}\}$:

$$\nabla\psi(y) = \frac{2}{y_{n+1}^2 - y_1^2 - \cdots - y_n^2}\begin{bmatrix} -y_1 \\ \vdots \\ -y_n \\ y_{n+1} \end{bmatrix}, \qquad y^T\nabla\psi(y) = 2$$



(from Boyd & Vandenberghe, *Convex Optimization*)

## Central path

- for $t > 0$, define $x^\star(t)$ as the solution of

$$\begin{array}{ll} \text{minimize} & tf_0(x) + \phi(x) \\ \text{subject to} & Ax = b \end{array}$$

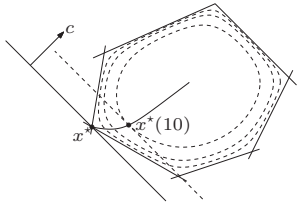
  (for now, assume $x^\star(t)$ exists and is unique for each $t > 0$)

- central path is $\{x^\star(t) \mid t > 0\}$

**example:** central path for an LP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x \le b_i, \quad i = 1, \ldots, 6 \end{array}$$

hyperplane $c^T x = c^T x^\star(t)$ is tangent to level curve of $\phi$ through $x^\star(t)$




- In practice, $t$ may increase to a very large number. But for each $t$, we only need to carry one or a few iterations. (Damped Newton steps for example)

Log-barrier formulation:

$$\min\{tf_0(\mathbf{x}) + \phi(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

Complexity of log-barrier interior-point method:

$$k \sim \left\lceil \frac{\log((\sum_i \theta_i)/(\varepsilon t^{(0)}))}{\log \mu} \right\rceil$$

- This is a polynomial time algorithm.

## Primal-dual interior-point methods

more efficient than barrier method when high accuracy is needed

- update primal and dual variables at each iteration; no distinction between inner and outer iterations
- often exhibit superlinear asymptotic convergence
- search directions can be interpreted as Newton directions for modified KKT conditions
- can start at infeasible points
- cost per iteration same as barrier method

- For students who can not fully understand this method, you may remember that it is a very good method and use it when you have options.

# $\ell_1$ minimization by interior-point method

Model

$$\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \le \sigma\}$$

$\Leftrightarrow$

$$\min_{\mathbf{x}} \min_{\mathbf{x}_1,\mathbf{x}_2}\{\mathbf{1}^T(\mathbf{x}_1+\mathbf{x}_2) : \mathbf{x}_1 \ge \mathbf{0}, \mathbf{x}_2 \ge \mathbf{0}, \mathbf{x}_1-\mathbf{x}_2 = \mathbf{x}, \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \le \sigma\}$$

$\Leftrightarrow$

$$\min_{\mathbf{x}_1,\mathbf{x}_2}\{\mathbf{1}^T(\mathbf{x}_1+\mathbf{x}_2) : \mathbf{x}_1 \ge \mathbf{0}, \mathbf{x}_2 \ge \mathbf{0}, \|\mathbf{A}(\mathbf{x}_1-\mathbf{x}_2)-\mathbf{b}\|_2 \le \sigma\}$$

$\Leftrightarrow$

$$\min_{\mathbf{x}_1,\mathbf{x}_2}\{\mathbf{1}^T\mathbf{x}_1+\mathbf{1}^T\mathbf{x}_2 : \mathbf{A}\mathbf{x}_1-\mathbf{A}\mathbf{x}_2+\mathbf{y}=\mathbf{b}, z=\sigma, (\mathbf{x}_1,\mathbf{x}_2,z,\mathbf{y}) \succeq_{\mathcal{K}} \mathbf{0}\}$$

where $(\mathbf{x}_1,\mathbf{x}_2,z,\mathbf{y}) \succeq_{\mathcal{K}} \mathbf{0}$ means

- $\mathbf{x}_1,\mathbf{x}_2 \in \mathbb{R}^n_+$,
- $(t,\mathbf{y}) \in \mathcal{Q}^{m+1}$.

Solver: Mosek, SDPT3, Gurobi.

Also, modeling language CVX and YALMIP.





- The first step of derivation uses the fact that:

$$\|\mathbf{x}\|_1 = \min_{\mathbf{x}_1,\mathbf{x}_2}\{\mathbf{1}^T(\mathbf{x}_1+\mathbf{x}_2) : \mathbf{x}_1-\mathbf{x}_2 = \mathbf{x}, \mathbf{x}_1 \ge \mathbf{0}, \mathbf{x}_2 \ge \mathbf{0}\}.$$

## Nuclear-norm minimization by interior-point method

If we can model

$$\min_{\mathbf{X}} \{\|\mathbf{X}\|_* : \mathcal{A}(\mathbf{X}) = \mathbf{b}\} \tag{9}$$

as an SDP ... (how? see next slide) ...

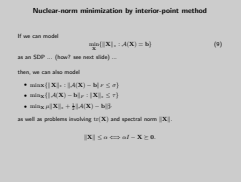
then, we can also model

- $\min_{\mathbf{X}} \{\|\mathbf{X}\|_* : \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|_F \leq \sigma\}$
- $\min_{\mathbf{X}} \{\|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|_F : \|\mathbf{X}\|_* \leq \tau\}$
- $\min_{\mathbf{X}} \mu\|\mathbf{X}\|_* + \frac{1}{2}\|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|_F^2$

as well as problems involving $\mathrm{tr}(\mathbf{X})$ and spectral norm $\|\mathbf{X}\|$.

$$\|\mathbf{X}\| \leq \alpha \Longleftrightarrow \alpha I - \mathbf{X} \succeq \mathbf{0}.$$






The proof of the last statement:

- $\Rightarrow$ Since $\mathbf{X}$ is positive semidefinite, $\|\mathbf{X}\|_2$ is the largest eigenvalue of $\mathbf{X}$. Therefore, $\alpha I - \mathbf{X}$ is positive semidefinite.
- $\Leftarrow$ Let $\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ be the spectral decomposition of $\mathbf{X}$. $\mathbf{V}$ is orthogonal and $\mathbf{D}$ is diagonal with all the eigenvalues $D_{ii}$ of $\mathbf{X}$ on the diagonal. $\alpha I - \mathbf{X} \succeq \mathbf{0} \Leftrightarrow \alpha I - \mathbf{D} \succeq \mathbf{0} \Leftrightarrow \alpha - D_{ii} \geq 0, \ \forall i$. Therefore, the largest eigenvalue of $\mathbf{X}$ is not larger than $\alpha$, i.e. $\|X\| \leq \alpha$.

## Sparse calculus for $\ell_1$

- inspect $|x|$ to get some ideas:
  $y, z \geq 0$ and $\sqrt{yz} \geq |x| \Longrightarrow \frac{1}{2}(y+z) \geq \sqrt{yz} \geq |x|$.
  moreover, $\frac{1}{2}(y+z) = \sqrt{yz} = |x|$ if $y = z = |x|$.
- observe

$$y, z \geq 0 \text{ and } \sqrt{yz} \geq |x| \Longleftrightarrow \begin{bmatrix} y & x \\ x & z \end{bmatrix} \succeq \mathbf{0}.$$

  So,

$$\begin{bmatrix} y & x \\ x & z \end{bmatrix} \succeq \mathbf{0} \Longrightarrow \frac{1}{2}(y+z) \geq |x|.$$

- we attain $\frac{1}{2}(y+z) = |x|$ if $y = z = |x|$.

Therefore, given $x$, we have

$$|x| = \min_{\mathbf{M}} \left\{ \frac{1}{2}\mathrm{tr}(\mathbf{M}) : \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \bullet \mathbf{M} = x, \mathbf{M} = \mathbf{M}^T, \mathbf{M} \succeq \mathbf{0} \right\}.$$

## Generalization to nuclear norm

- Consider $\mathbf{X} \in \mathbb{R}^{m\times n}$ (w.o.l.g., assume $m \le n$) and let's try imposing

$$\begin{bmatrix} \mathbf{Y} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{Z} \end{bmatrix} \succeq \mathbf{0}$$

- Diagonalize $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$, $\|\mathbf{X}\|_* = \sum_i \sigma_i$.

$$[\mathbf{U}^T, -\mathbf{V}^T] \begin{bmatrix} \mathbf{Y} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ -\mathbf{V} \end{bmatrix} = \mathbf{U}^T\mathbf{Y}\mathbf{U} + \mathbf{V}^T\mathbf{Z}\mathbf{V} - \mathbf{U}^T\mathbf{X}\mathbf{V} - \mathbf{V}^T\mathbf{X}^T\mathbf{U}$$
$$= \mathbf{U}^T\mathbf{Y}\mathbf{U} + \mathbf{V}^T\mathbf{Z}\mathbf{V} - 2\Sigma \ \succeq \mathbf{0}.$$

So, $\mathrm{tr}(\mathbf{U}\mathbf{Y}\mathbf{U}^T + \mathbf{V}\mathbf{Z}\mathbf{V}^T - 2\Sigma) = \mathrm{tr}(\mathbf{Y}) + \mathrm{tr}(\mathbf{Z}) - 2\|\mathbf{X}\|_* \ge 0$.

- To attain "=", we can let $\mathbf{Y} = \mathbf{U}\Sigma\mathbf{U}^T$ and $\mathbf{Z} = \mathbf{V}\Sigma_{n\times n}\mathbf{V}^T$.

- This derivation mimics the scalar case on the previous slide. The idea is that if one imposes the constraint $\begin{bmatrix} \mathbf{Y} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{Z} \end{bmatrix} \succeq \mathbf{0}$, then $\frac{1}{2}\left(\mathrm{tr}(\mathbf{Y}) + \mathrm{tr}(\mathbf{Z})\right) \ge \|\mathbf{X}\|_*$ and $=$ can be attained.
- The singular value decomposition (SVD) of a real matrix $\mathbf{X}$: $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$,where $\mathbf{U}^T\mathbf{U} = I$,$\mathbf{V}^T\mathbf{V} = I$, $\Sigma$ is diagonal with all the singular values of $\mathbf{X}$ on the diagonal.
- Following its definition, $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$.

Therefore,

$$\|\mathbf{X}\|_* = \min_{\mathbf{Y},\mathbf{Z}} \left\{ \frac{1}{2}(\mathrm{tr}(\mathbf{Y}) + \mathrm{tr}(\mathbf{Z})) : \begin{bmatrix} \mathbf{Y} & \mathbf{X} \\ \mathbf{X}^T & \mathbf{Z} \end{bmatrix} \succeq \mathbf{0} \right\} \tag{10}$$

$$= \min_{\mathbf{M}} \left\{ \frac{1}{2}\mathrm{tr}(\mathbf{M}) : \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \bullet \mathbf{M} = \mathbf{X}, \mathbf{M} = \mathbf{M}^T, \mathbf{M} \succeq \mathbf{0} \right\}. \tag{11}$$

Exercise: express the following problems as SDPs

- $\min_{\mathbf{X}}\{\|\mathbf{X}\|_* : \mathcal{A}(\mathbf{X}) = \mathbf{b}\}$
- $\min_{\mathbf{X}} \mu\|\mathbf{X}\|_* + \frac{1}{2}\|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|_F$
- $\min_{\mathbf{L},\mathbf{S}}\{\|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 : \mathcal{A}(\mathbf{L}+\mathbf{S}) = \mathbf{b}\}$

- The reason to reformulate (10) into (11) is that: in (10), the positive semidesert matrix $\begin{pmatrix} \mathbf{Y} & \mathbf{X} \\ \mathbf{X^T} & \mathbf{Z} \end{pmatrix}$ is represented as a block matrix in which $\mathbf{Y}$ and $\mathbf{Z}$ are free but $\mathbf{X}$ is given. It does not form a standard SDP, which is composed of positive semi-definite matrices and their linear objective and constraints. In (11), $\mathbf{M}$ is the unknown matrix. The first constraint together with $\mathbf{M} = \mathbf{M}^T$ make its off-diagonal block equal to the given matrix $\mathbf{X}$.

## Practice of interior-point methods (IPMs)

- LP, SOCP, SDP are well known and have reliable (commercial, off-the-shelf) solvers
- Yet, the most reliable solvers cannot handle large-scale problems (e.g., images, video, manifold learning, distributed stuff, ...)
  - Example: to recover a still image, there can be 10M variables and 1M constraints. Even worse, the constraint coefficients are dense. Result: Out of memory.
- Simplex and active-set methods: matrix containing $\mathbf{A}$ must be inverted or factorized to compute the next point (unless $\mathbf{A}$ is very sparse).
- IPMs approximately solve a Newton system and thus also factorize a matrix involving $\mathbf{A}$.
- Even large and dense matrices can be handled, for sparse optimization, one should take advantages of the solution sparsity.
- Some compressive sensing problems have $\mathbf{A}$ with structures friendly for operations like $\mathbf{Ax}$ and $\mathbf{A}^T\mathbf{y}$.





Practice of interior-point methods (IPMs)
- LP, SOCP, SDP are well known and have reliable (commercial, off-the-shelf) solvers
- Yet, the most reliable solvers cannot handle large-scale problems (e.g., images, video, manifold learning, distributed stuff, ...)
  - Example: to recover a still image, there can be 10M variables and 1M constraints. Even worse, the constraint coefficients are dense. Result: Out of memory.
- Simplex and active-set methods: matrix containing A must be inverted or factorized to compute the next point (unless A is very sparse).
- IPMs approximately solve a Newton system and thus also factorize a matrix involving A.
- Even large and dense matrices can be handled, for sparse optimization, one should take advantages of the solution sparsity.
- Some compressive sensing problems have A with structures friendly for operations like Ax and $A^Ty$.

## Practice of interior-point methods (IPMs)

- The Simplex, active-set, and IPMs have *reliable* solvers; good to be the benchmark.
- They have nice interfaces (including *CVX* and *YALMIP*, which save you time.)
  *CVX* and *YALMIP* are not solvers; they translate problems and then call solvers; see `http://goo.gl/zUlMK` and `http://goo.gl/1u0xP`.
- They can return *highly accurate* solutions; some first-order algorithms (coming later in this course) do not always.
- There are other remedies; see next slide.



Practice of interior-point methods (IPMs)
- The Simplex, active-set, and IPMs have reliable solvers; good to be the benchmark.
- They have nice interfaces (including CVX and YALMIP, which save you time.)
  CVX and YALMIP are not solvers; they translate problems and then call solvers; see http://goo.gl/zUlMK and http://goo.gl/1u0xP.
- They can return highly accurate solutions; some first-order algorithms (coming later in this course) do not always.
- There are other remedies; see next slide.

# Papers of large-scale SDPs

- Low-rank factorizations:
  - S. Burer and R. D. C. Monteiro, A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization, Math. Program., 95:329–357, 2003.
  - LMaFit, `http://lmafit.blogs.rice.edu/`
- First-order methods for conic programming:
  - Z. Wen, D. Goldfarb, and W. Yin. Alternating direction augmented Lagrangian methods for semidefinite programming. Math. Program. Comput., 2(3-4):203–230, 2010.
- Matrix-free IPMs:
  - K. Fountoulakis, J. Gondzio, P. Zhlobich. Matrix-free interior point method for compressed sensing problems, 2012. `http://www.maths.ed.ac.uk/~gondzio/reports/mfCS.html`





Papers of large-scale SDPs

- Low-rank factorizations:
  - S. Burer and R. D. C. Monteiro, A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization, Math. Program., 95:329–357, 2003.
  - LMaFit, http://lmafit.blogs.rice.edu/
- First-order methods for conic programming:
  - Z. Wen, D. Goldfarb, and W. Yin. Alternating direction augmented Lagrangian methods for semidefinite programming. Math. Program. Comput., 2(3-4):203–230, 2010.
- Matrix-free IPMs:
  - K. Fountoulakis, J. Gondzio, P. Zhlobich. Matrix-free interior point method for compressed sensing problems, 2012. http://www.maths.ed.ac.uk/~gondzio/reports/mfCS.html

## Subgradient methods

Sparse optimization is typically nonsmooth, so it is natural to consider subgradient methods.

- apply subgradient descent to, say, $\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$.
- apply projected subgradient descent to, say, $\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$.

Good: subgradients are easy to derive, methods are simple to implement.

Bad: convergence requires carefully chosen step sizes (classical ones require diminishing step sizes). Convergence rate is weak on paper (and in practice, too?)

Further readings: `http://arxiv.org/pdf/1001.4387.pdf`, `http://goo.gl/qFVA6`, `http://goo.gl/vC21a`.





Subgradient methods

Sparse optimization is typically nonsmooth, so it is natural to consider subgradient methods.

- apply subgradient descent to, say, $\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$.
- apply projected subgradient descent to, say, $\min_{\mathbf{x}}\{\|\mathbf{x}\|_1 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$.

Good: subgradients are easy to derive, methods are simple to implement.

Bad: convergence requires carefully chosen step sizes (classical ones require diminishing step sizes). Convergence rate is weak on paper (and in practice, too?)

Further readings: `http://arxiv.org/pdf/1001.4387.pdf`, `http://goo.gl/qFVA6`, `http://goo.gl/vC21a`.