18 Lecture 18, Mar 3

Announcements

- HW5 (Newton's method, handwritten digit recognition) due today @ 11:59PM.
- HW6 (EM/MM, handwritten digit recognition revisited) posted. Due next Fri Mar 11 @ 11:59PM.
- Solution sketches for HW1-4 are posted. `http://hua-zhou.github.io/teaching/biostatm280-2016winter/hwXXsol.html`. Substitute XX by 01, 02, ...

  Solution sketch for HW5 will be posted tomorrow after the due time.
- Quiz 3 returned. You did great job.
- Quiz 4 next Thu in class.
- Don't forget course evaluation: `http://my.ucla.edu`.

Last time

- Quasi-Newton method. Users only need to input functions for evaluating objective values and gradients.
- Conjugate gradient (CG) method for solving huge, sparse or structured linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.
- Pre-conditioned CG for improving the convergence of CG.

Today

- Nonlinear conjugate gradient.
- Convex optimization: introduction.

More buzzwords and softwares

Here are a few variants of CG that we should at least know the names and what they are for (so we can Google later in need).

- MINRES (minimum residual method): symmetric indefinite $\boldsymbol{A}$.
- Bi-CG (bi-conjugate gradient): unsymmetric $\boldsymbol{A}$.
- Bi-CGSTAB (Bi-CG stabilized): improved version of Bi-CG.
- GMRES (generalized minimum residual method): current *de facto* method for unsymmetric $\boldsymbol{A}$. E.g., PageRank problem.
- Lanczos method: top eigen-pairs of a large symmetric matrix.
- Arnoldi method: top eigen-pairs of a large unsymmetric matrix.
- Lanczos bidiagonalization algorithm: top singular triplets of large matrix.

  Remark: For Lanczos/Arnoldi methods, the critical computation is still matrix vector multiplication $\boldsymbol{Av}$.

Softwares:

- MATLAB:
  - Iterative methods for solving linear equations:
    `pcg`, `bicg`, `bicgstab`, `gmres`, ...
  - Iterative methods for top eigen-pairs and singular pairs:
    `eigs`, `svds`, ...
  - Pre-conditioner:
    `cholinc`, `luinc`, ...

  Get familiar with the reverse communication interface (RCI) for utilizing iterative solvers:

```
x = gmres(A, b)
x = gmres(@Afun, b)
eigs(A)
eigs(@Afun)
```

- Consider the PageRank problem. We want to find the top left eigenvector of the transition matrix

$$\boldsymbol{P} = p\boldsymbol{R}^{+}\boldsymbol{A} + \boldsymbol{z}\boldsymbol{1}_n^{\intercal},$$

where $\boldsymbol{R} = \text{diag}(r_1, \ldots, r_n)$ and $z_j = (1-p)/n$ if $r_i > 0$ and $1/n$ if $r_i = 0$. Size of $\boldsymbol{P}$ can be huge: $n \approx 40$ billion web pages. How to call the `gmres` or `eigs` function?

- R: Try Google and good luck ...
- JULIA. `IterativeSolvers.jl` package. `https://github.com/JuliaLang/IterativeSolvers.jl/issues/1`

(Nonlinear) Conjugate gradient method

- *Linear* conjugate gradient method is for solving linear system $\boldsymbol{Ax} = \boldsymbol{b}$, or equivalently, minimizing $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Ax} - \boldsymbol{b}^T\boldsymbol{x}$.
- *Nonlinear* conjugate gradient is for nonlinear optimization

$$\text{minimize } f(\boldsymbol{x}).$$

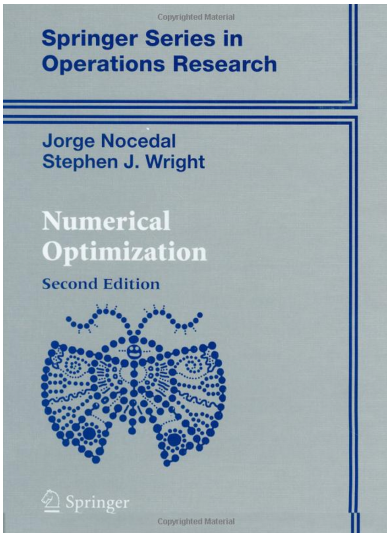
- History: Fletcher and Reeves in 60s.
- Fletcher-Reeves CG for nonlinear minimization:

  Given $\boldsymbol{x}^{(0)}$
  Evaluate $\nabla f^{(0)} = \nabla f(\boldsymbol{x}^{(0)})$
  Set $\boldsymbol{p}^{(0)} \leftarrow -\nabla f^{(0)}$, $t \leftarrow 0$
  while $\nabla f^{(t)} \neq \boldsymbol{0}$ **do**
    Compute $\alpha^{(t)}$ and set $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \alpha^{(t)}\boldsymbol{p}^{(t)}$
    Evaluate $\nabla f^{(t+1)} = \nabla f(\boldsymbol{x}^{(t+1)})$
    $\beta^{(t+1)} \leftarrow \frac{df^{(t+1)}\nabla f^{(t+1)}}{df^{(t)}\nabla f^{(t)}}$
    $\boldsymbol{p}^{(t+1)} \leftarrow -\nabla f^{(t+1)} + \beta^{(t+1)}\boldsymbol{p}^{(t)}$
    $t \leftarrow t + 1$
  end while

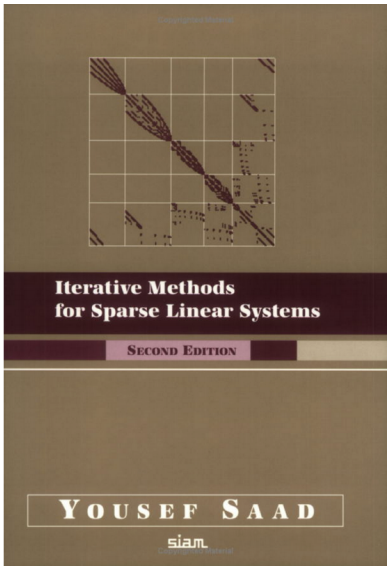
- Most cost is evaluation of objective function and its gradient. No matrix operations are needed. Appealing for large nonlinear optimization problems.
- Line search (choose $\alpha^{(t)}$) is necessary to get a descending algorithm.
- Nonlinear conjugate gradient is implemented in the `optim()` function in R. Only for unconstrained problems.

Reference books

Newton's method for constrained problems, quasi-Newton, conjugate gradient, and many more:



Krylov subspace methods:



Some remarks on optimization

> There is simply no such thing as a universal 'gold standard' when it comes to algorithms.
>
> Unknown Reviewer

- Nonlinear optimization algorithms

Algorithm	Convergence Rate	Per-iteration Cost	Example
Newton	quadratic	high, usually $O(np^2) + O(p^3)$	GLM with canonical link
Fisher Scoring	super-linear	high, usually $O(np^2) + O(p^3)$	GLM with non-canonical link
Gauss-Newton	super-linear	high, usually $O(np^2) + O(p^3)$	nonlinear GLM
Quasi-Newton	super-linear	moderate, usually $O(np) + O(p^2)$	
Conjugate gradient	super-linear	moderate, usually $O(np) + O(p^2)$	
Coordinate descent	linear	low	
Steepest descent	linear	low	
EM/MM	linear	low	

- *Problem specific analysis* is critical for developing a successful optimization algorithm.

  E.g., I don't think any black-box procedure can beat our safe-guarded Newton's method for the Dirichlet-Multinomial MLE problem (HW5/6) in terms of efficiency.

- Use "black-box" for
  - numerical linear algebra
  - convex programming (TODO today and next week). Current "technology" (`Gurobi`, `Mosek`, `Cplex`, `cvx`, `Matlab`, ...) can deal with problems with up to $10^3 \sim 10^4$ variables and constraints or even more with structure.

- First order methods (EM/MM, CD, steepest descent) is easier for parallel computing.

> Algorithm development goes hand in glove with hardware advancement.
>
> Hua

- Reference books:
  - *Numerical Optimization* (Nocedal and Wright, 2006)
  - *Convex Optimization* (Boyd and Vandenberghe, 2004)
  - *The EM algorithm and Extensions* (McLachlan and Krishnan, 2008)
  - *Numerical Analysis for Statisticians* (Lange, 2010)

Convex optimization problems

- A *mathematical optimization problem*, or just *optimization problem*, has the form

$$\begin{array}{ll} \text{minimize} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \le b_i, \quad i = 1, \ldots, m. \end{array}$$

  Here $f_0 : \mathbf{R}^n \mapsto \mathbf{R}$ is the *objective* function and $f_i : \mathbf{R}^n \mapsto \mathbf{R}$, $i = 1, \ldots, m$, are the *constraint* functions.

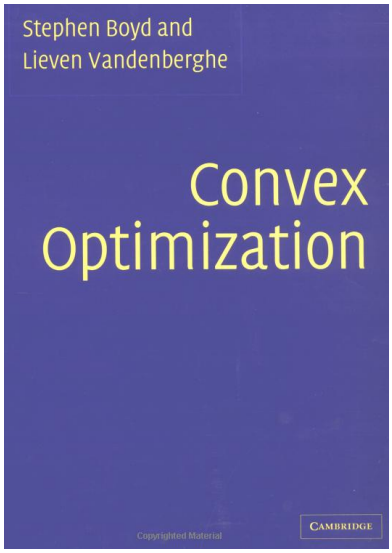
  ☞ An equality constraint $f_i(\boldsymbol{x}) = b_i$ can be absorbed into inequality constraints $f_i(\boldsymbol{x}) \le b_i$ and $-f_i(\boldsymbol{x}) \le -b_i$.

- If the objective and constraint functions are convex, then it is called a *convex optimization problem*.

  ☞ In a convex optimization problem, only linear equality constraint of form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is allowed (why?).

- Convex optimization is becoming a *technology*. Therefore it is important to recognize, formulate, and solve convex optimization problems.

- A definite resource is the book *Convex Optimization* by Boyd and Vandenberghe, which is freely available at `http://stanford.edu/~boyd/cvxbook/`. Same website has links to slides, code, and lecture videos.



Lecture videos:
`http://www.stanford.edu/class/ee364a/videos.html`
`http://www.stanford.edu/class/ee364b/videos.html`

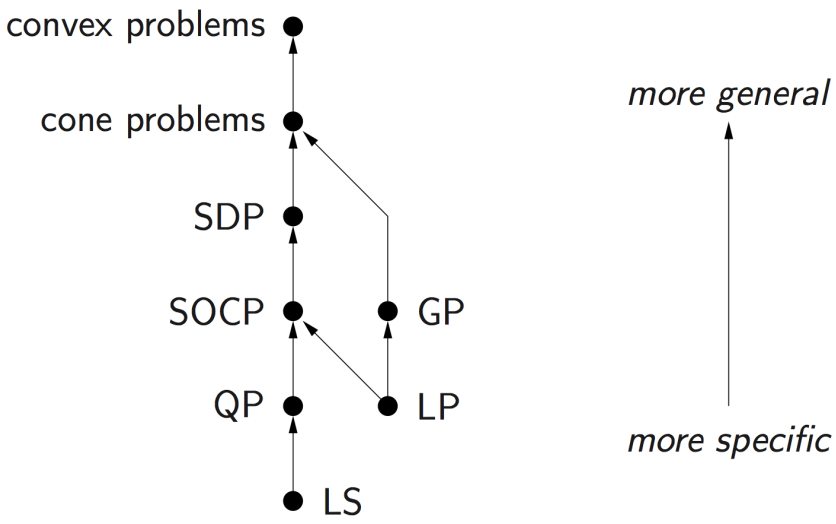
- UCLA courses by Lieven Vandenberghe: EE236A (Linear Programming), EE236B (Convex Optimization), EE236C (Optimization Methods for Large-scale Systems).

- Convex programming (LS, LP, QP, GP, SOCP, SDP) is almost becoming a technology (`Cplex`, `Gurobi`, `Mosek`, `cvx`, `Matlab`, `JuliaOpt`, ...), somewhat like numerical linear algebra libraries BLAS and LAPACK. Current technology can solve convex problems with up to thousands of variables and constraints.

- In this course, we learn basic terminology and how to recognize and solve some standard convex programming problems.

Hierarchy of convex optimization problems

We have spent a fair amount of time on the LS (least squares) problem. In the next couple of lectures, we study LP (linear programming), QP (quadratic programming), SOCP (second-order cone programming), SDP (semidefinite programming), and GP (geometric programming), with an emphasis on statistical applications and software implementation.



Optimization softwares

Like computer languages, getting familiar with *good* optimization softwares broadens the scope and scale of problems we are able to solve in statistics.

- Following table lists some of the best convex optimization softwares. Use of modeling tools `cvx` (for MATLAB) or `Convex.jl` (for JULIA, coupled with solvers Mosek and/or Gurobi, is highly recommended.

  ☞ Gurobi is named after its founders: Zonghao **Gu**, Edward **Ro**thberg, and Robert **Bi**xby. Bixby founded the CPLEX at IBM, while Rothberg and Gu led the CPLEX development team for nearly a decade.

- Difference between *modeling tool* and *solvers*.

  - Solvers (`Gurobi`, `Mosek`, `Cplex`, ...) are concrete software implementation of optimization algorithms.
  - Modeling tools such as `cvx` (for MATLAB) and `Convex.jl` (Julia analog of cvx) implement the disciplined convex programming (DCP) paradigm proposed by Grant and Boyd (2008). `http://stanford.edu/~boyd/papers/disc_cvx_prog.html`. DCP prescribes a set of simple rules from which users can construct convex optimization problems easily.

Modeling tools usually have the capability to use a variety of solvers. But modeling tools are solver agnostic so users do not have to worry about specific solver interface.

- In this course, I focus more on using the `Convex.jl` package in Julia. For using `cvx` for Matlab, take Boyd or Vandenberghe's class. Also check Brian Gaines slides `http://brgaines.github.io/talks/cvx/cvxDemo.html`.

	LP	MILP	SOCP	MISOCP	SDP	GP	NLP	MINLP	R	Matlab	Julia	Python	Cost
JuMP.jl	✓	✓	✓	✓			✓	✓			✓		O
Convex.jl	✓	✓	✓	✓	✓						✓		O
cvx	✓	✓	✓	✓	✓	✓				✓		✓	A
Gurobi	✓	✓	✓	✓					✓	✓	✓	✓	A
Mosek	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	A
CPLEX	✓	✓	✓						?	✓	✓	✓	A
SCS	✓		✓		✓					✓	✓	✓	O
SeDuMi	✓		✓		✓	?				✓			O
SDPT3	✓		✓		✓	?				✓			O
KNITRO	✓	✓					✓	✓		✓	✓	✓	$

LP = Linear Programming, MILP = Mixed Integer LP, SOCP = Second-order cone programming (includes QP, QCQP), MISOCP = Mixed Integer SOCP, SDP = Semidefinite Programming, GP = Geometric Programming, NLP = (constrained) Nonlinear Programming (includes general QP, QCQP), MINLP = Mixed Integer NLP, O = Open source, A = Free academic license

Set up Gurobi on your computer

1. Register for an account on `http://www.gurobi.com`. Be sure to use your `edu` email and check `Academic` as your account type.
2. After confirmation of your academic account, log into your account and request a free academic license at `http://www.gurobi.com/download/licenses/free-academic`.
3. Run `grbgetkey` command on command line and enter the key you obtained in step 2.
4. Now you should be able to use Gurobi in Matlab, R, and Julia.

Set up Mosek on your computer

Follow instructions at `https://www.mosek.com/resources/academic-license`

Set up CVX on your computer

1. Request a free academic (professional) license at `http://cvxr.com/cvx/academic/` using your `edu` email. Your will receive the license file `license.dat` by email.

2. Within Matlab, type
   `cvx_setup /path/cvx_license.dat`

3. Now you should be able to use CVX in Matlab.

   ☞ The *standard license* comes with free solvers `SeDuMi` and `SDPT3`. The *Academic license* also bundles with `Gurobi` and `Mosek`.

Linear programming (LP)

- A general linear program takes the form

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h}. \end{array}$$

  Linear program is a convex optimization problem, why?

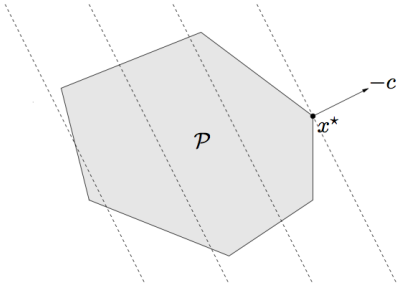


Figure 4.4 Geometric interpretation of an LP. The feasible set $\mathcal{P}$, which is a polyhedron, is shaded. The objective $c^T x$ is linear, so its level curves are hyperplanes orthogonal to c (shown as dashed lines). The point $x^\star$ is optimal; it is the point in $\mathcal{P}$ as far as possible in the direction $-c$.

- The *standard form* of an LP is

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \succeq \boldsymbol{0}. \end{aligned}$$

  To transform a general linear program into the standard form, we introduce the *slack variables* $\boldsymbol{s} \succeq \boldsymbol{0}$ such that $\boldsymbol{G}\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{h}$. Then we write $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^-$, where $\boldsymbol{x}^+ \succeq \boldsymbol{0}$ and $\boldsymbol{x}^- \succeq \boldsymbol{0}$. This yields the problem

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T(\boldsymbol{x}^+ - \boldsymbol{x}^-) \\ \text{subject to} \quad & \boldsymbol{A}(\boldsymbol{x}^+ - \boldsymbol{x}^-) = \boldsymbol{b} \\ & \boldsymbol{G}(\boldsymbol{x}^+ - \boldsymbol{x}^-) + \boldsymbol{s} = \boldsymbol{h} \\ & \boldsymbol{x}^+ \succeq \boldsymbol{0}, \boldsymbol{x}^- \succeq \boldsymbol{0}, \boldsymbol{s} \succeq \boldsymbol{0} \end{aligned}$$

  in $\boldsymbol{x}^+$, $\boldsymbol{x}^-$, and $\boldsymbol{s}$.

  ☞ Slack variables are often used to transform a complicated inequality constraint to simple non-negativity constraints.

- The *inequality form* of an LP is

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h}. \end{aligned}$$

  ☞ Some softwares, e.g., `solveLP` in `R`, require an LP be written in either standard or inequality form. However a good software should do this for you!

- A *piecewise-linear minimization* problem

$$\text{minimize} \quad \max_{i=1,\ldots,m} (\boldsymbol{a}_i^T \boldsymbol{x} + b_i)$$

  can be transformed to an LP

$$\begin{aligned} \text{minimize} \quad & t \\ \text{subject to} \quad & \boldsymbol{a}_i^T \boldsymbol{x} + b_i \le t, \quad i = 1, \ldots, m, \end{aligned}$$

  in $\boldsymbol{x}$ and t. Apparently

$$\text{minimize} \max_{i=1,\ldots,m} |\boldsymbol{a}_i^T \boldsymbol{x} + b_i|$$

and

$$\text{minimize} \max_{i=1,\ldots,m} (\boldsymbol{a}_i^T \boldsymbol{x} + b_i)_+$$

are also LP.

☞ Any *convex optimization problem*

$$\begin{array}{ll} \text{minimize} & f_0(\boldsymbol{x}) \\ \text{subject to} & f_i(\boldsymbol{x}) \le 0, \quad i = 1, \ldots, m \\ & \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i = 1, \ldots, p, \end{array}$$

where $f_0, \ldots, f_m$ are convex functions, can be transformed to the *epigraph form*

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & f_0(\boldsymbol{x}) - t \le 0 \\ & f_i(\boldsymbol{x}) \le 0, \quad i = 1, \ldots, m \\ & \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i = 1, \ldots, p \end{array}$$

in variables $\boldsymbol{x}$ and t. That is why people often say linear program is universal.

- The *linear fractional programming*

$$\begin{array}{ll} \text{minimize} & \dfrac{\boldsymbol{c}^T \boldsymbol{x} + d}{\boldsymbol{e}^T \boldsymbol{x} + f} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \boldsymbol{e}^T \boldsymbol{x} + f > 0 \end{array}$$

can be transformed to an LP

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T \boldsymbol{y} + dz \\ \text{subject to} & \boldsymbol{G}\boldsymbol{y} - z\boldsymbol{h} \preceq \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{y} - z\boldsymbol{b} = \boldsymbol{0} \\ & \boldsymbol{e}^T \boldsymbol{y} + fz = 1 \\ & z \ge 0 \end{array}$$

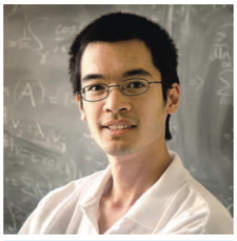
in $\boldsymbol{y}$ and z, via transformation of variables

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{\boldsymbol{e}^T \boldsymbol{x} + f}, \quad z = \frac{d}{\boldsymbol{e}^T \boldsymbol{x} + f}.$$

See Boyd and Vandenberghe (2004, Section 4.3.2) for proof.

- LP Example. Compressed sensing (Candès and Tao, 2006; Donoho, 2006) tries to address a fundamental question: how to compress and transmit a complex signal (e.g., musical clips, mega-pixel images), which can be decoded to recover the original signal?

Emmanuel Candes. *(Photo courtesy of Emmanuel Candes.)*



Terence Tao. *(Photo courtesy of Reed Hutchinson/UCLA.)*

Suppose a signal $\boldsymbol{x} \in \mathbf{R}^n$ is sparse with s non-zeros. We under-sample the signal by multiplying a (flat) measurement matrix $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{A} \in \mathbf{R}^{m \times n}$ has iid normal entries. Candès et al. (2006) show that the solution to

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{x}\|_1 \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{aligned}$$

exactly recovers the true signal under certain conditions on $\boldsymbol{A}$ when $n \gg s$ and $m \approx s \ln(n/s)$. Why sparsity is a reasonable assumption? *Virtually all real-world images have low information content.*

The ℓ_1 minimization problem apparently is an LP, by writing $\boldsymbol{x} = \boldsymbol{x}^+ - \boldsymbol{x}^-$,

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^T(\boldsymbol{x}^+ + \boldsymbol{x}^-) \\ \text{subject to} \quad & \boldsymbol{A}(\boldsymbol{x}^+ - \boldsymbol{x}^-) = \boldsymbol{y} \\ & \boldsymbol{x}^+ \succeq \mathbf{0}, \boldsymbol{x}^- \succeq \mathbf{0}. \end{aligned}$$

Let's work on a numerical example. `http://hua-zhou.github.io/teaching/st790-2015spr/demo_cs.html`

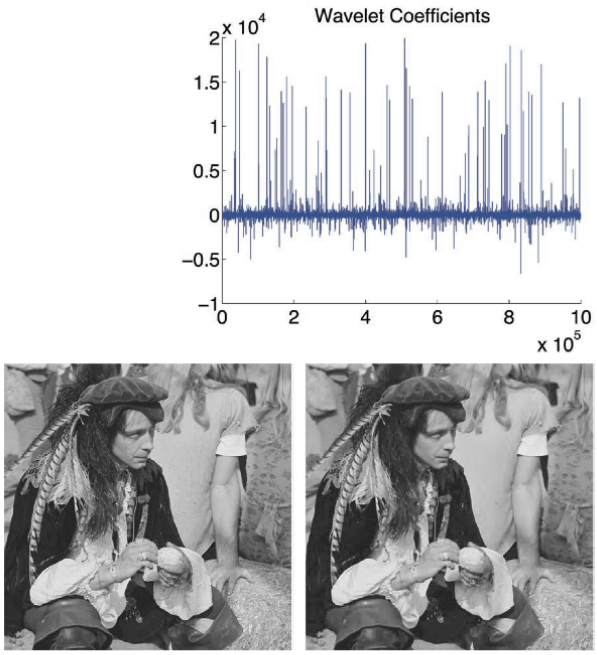
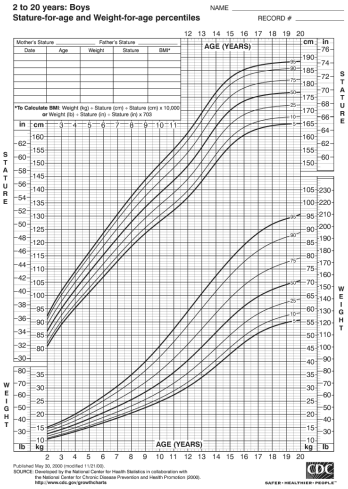
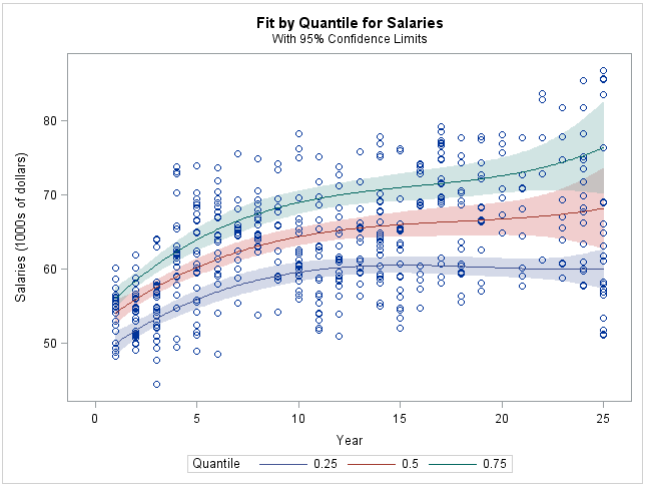


Figure 1. *Normal scenes from everyday life are compressible with respect to a basis of wavelets. (left) A test image. (top) One standard compression procedure is to represent the image as a sum of wavelets. Here, the coefficients of the wavelets are plotted, with large coefficients identifying wavelets that make a significant contribution to the image (such as identifying an edge or a texture). (right) When the wavelets with small coefficients are discarded and the image is reconstructed from only the remaining wavelets, it is nearly indistinguishable from the original. (Photos and figure courtesy of Emmanuel Candes.)*

- LP Example. Quantile regression. In linear regression, we model the mean of response variable as a function of covariates. In many situations, the error variance is not constant, the distribution of y may be asymmetric, or we simply care about the quantile(s) of response variable. Quantile regression offers a better modeling tool in these applications.





In τ-quantile regression, we minimize the loss function

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta}),$$

where $\rho_\tau(z) = z(\tau - 1_{\{z<0\}})$. Writing $\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{r}^+ - \boldsymbol{r}^-$, this is equivalent to the LP

$$\begin{array}{ll} \text{minimize} & \tau \mathbf{1}^T\boldsymbol{r}^+ + (1-\tau)\mathbf{1}^T\boldsymbol{r}^- \\ \text{subject to} & \boldsymbol{r}^+ - \boldsymbol{r}^- = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \\ & \boldsymbol{r}^+ \succeq \mathbf{0}, \boldsymbol{r}^- \succeq \mathbf{0} \end{array}$$

in $\boldsymbol{r}^+$, $\boldsymbol{r}^-$, and $\boldsymbol{\beta}$.

- LP Example: ℓ_1 regression. A popular method in robust statistics is the median absolute deviation (MAD) regression that minimizes the ℓ_1 norm of the residual vector $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_1$. This apparently is equivalent to the LP

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T(\boldsymbol{r}^+ + \boldsymbol{r}^-) \\ \text{subject to} & \boldsymbol{r}^+ - \boldsymbol{r}^- = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \\ & \boldsymbol{r}^+ \succeq \mathbf{0}, \boldsymbol{r}^- \succeq \mathbf{0} \end{array}$$

in $\boldsymbol{r}^+$, $\boldsymbol{r}^-$, and $\boldsymbol{\beta}$.

☞ ℓ_1 regression = MAD = 1/2-quantile regression.

- LP Example: ℓ_∞ regression (Chebychev approximation). Minimizing the worst possible residual $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_\infty$ is equivalent to the LP

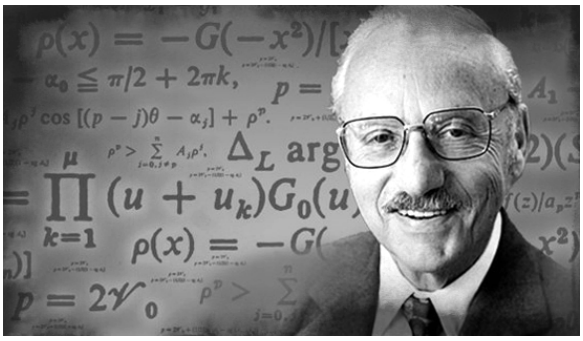
$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -t \le y_i - \boldsymbol{x}_i^T\boldsymbol{\beta} \le t, \quad i = 1, \ldots, n \end{array}$$

in variables $\boldsymbol{\beta}$ and t.

- LP Example: Dantzig selector. Candès and Tao (2007) propose a variable selection method called the Dantzig selector that solves

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\|_\infty \\ \text{subject to} & \sum_{j=2}^{p} |\boldsymbol{\beta}_j| \le t, \end{array}$$

which can be transformed to an LP. Indeed they name the method after George Dantzig, who invented the simplex method for efficiently solving LP in 50s.



☞ Apparently any loss/penalty or loss/constraint combinations of form

$$\{\ell_1, \ell_\infty, \text{quantile}\} \times \{\ell_1, \ell_\infty, \text{quantile}\},$$

possibly with affine (equality and/or inequality) constraints, can be formulated as an LP.

- Example: 1-norm SVM. In two-class classification problems, we are given training data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $\boldsymbol{x}_i \in \mathbf{R}^p$ are feature vectors and $y_i \in \{-1, 1\}$ are class labels. Zhu et al. (2004) propose the 1-norm support vector machine (svm) that achieves the dual purpose of classification and feature selection. Denote the solution of the optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \left[1 - y_i \left(\beta_0 + \sum_{j=1}^{p} x_{ij} \beta_j \right) \right]_+ \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j| \le t \end{aligned}$$

by $\hat{\beta}_0(t)$ and $\hat{\boldsymbol{\beta}}(t)$. 1-norm svm classifies a future feature vector $\boldsymbol{x}$ by the sign of fitted model

$$\hat{f}(\boldsymbol{x}) = \hat{\beta}_0 + \boldsymbol{x}^T \hat{\boldsymbol{\beta}}.$$

- Many more applications of LP: Airport scheduling (Copenhagen airport uses Gurobi), airline flight scheduling, NFL scheduling, `match.com`, LaTeX, ...