Disciplined Convex Optimization with CVXR

Anqi Fu Bala Narasimhan Stephen Boyd

EE & Statistics Departments

Stanford University

useR! Conference 2016

Outline

Convex Optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

with variable $x \in \mathbf{R}^n$

- Objective and inequality constraints $f_0, \ldots, f_m$ are convex
- Equality constraints are linear

Convex Optimization

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& Ax = b
\end{array}
$$

with variable $x \in \mathbf{R}^n$

- Objective and inequality constraints $f_0, \ldots, f_m$ are convex
- Equality constraints are linear

Why?

- We can solve convex optimization problems
- There are many applications in many fields

Outline

CVXR

A modeling language in R for convex optimization

- Open source down to the solvers
- Uses disciplined convex programming to verify convexity
- Supports parameters, multiple constraints
- Mixes easily with general R code and other libraries

Outline

Convex Optimization

CVXR

Code Examples

Disciplined Convex Programming

Future Work

Example: Ordinary Least Squares (OLS)

TODO: Mathematical definition of problem?

Example: Ordinary Least Squares (OLS)

```
library(cvxr)
beta <- Variable(n)
obj <- SumSquares(y - X %*% beta)
prob <- Problem(Minimize(obj))
solution <- solve(prob)
solution$opt_val
solution$beta
```

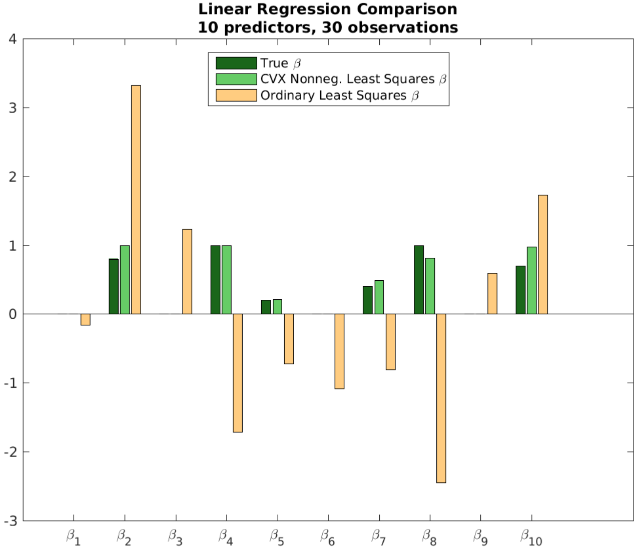
- `X` and `y` are constants; `beta`, `obj`, and `prob` are S4 objects
- `solve` method canonicalizes, solves, and returns a list with final objective and optimal value of each variable

Example: Non-Negative Least Squares

```
constr <- list(beta >= 0)
prob2 <- Problem(Minimize(obj), constr)
solution2 <- solve(prob2)
solution2$opt_val
solution2$beta
```

- Extend prior example by requiring `beta` be non-negative
- Construct new problem with list `constr` of constraints formed from constants and variables
- Variables, parameters, expressions, and constraints exist outside of any problem

Results: Non-Negative Least Squares



Overview: Huber Regression

$$\text{minimize} \quad \textstyle\sum_i^n \phi(y_i - \beta^T x_i)$$

with variable $\beta \in \mathbf{R}^n$ and Huber function

$$\phi(u) \quad = \begin{cases} u^2 & |u| \leq 0 \\ 2Mu - M^2 & |u| > 0 \end{cases}$$

where $M > 0$ is the threshold

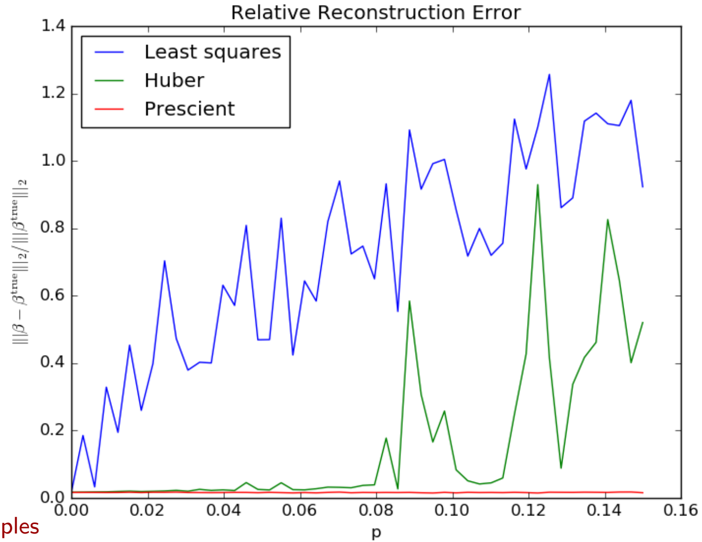
- TODO: Plot of Huber function?
- Same as OLS for small residuals, allows some large residuals
- Better fit when data contains outliers

Example: Huber Regression

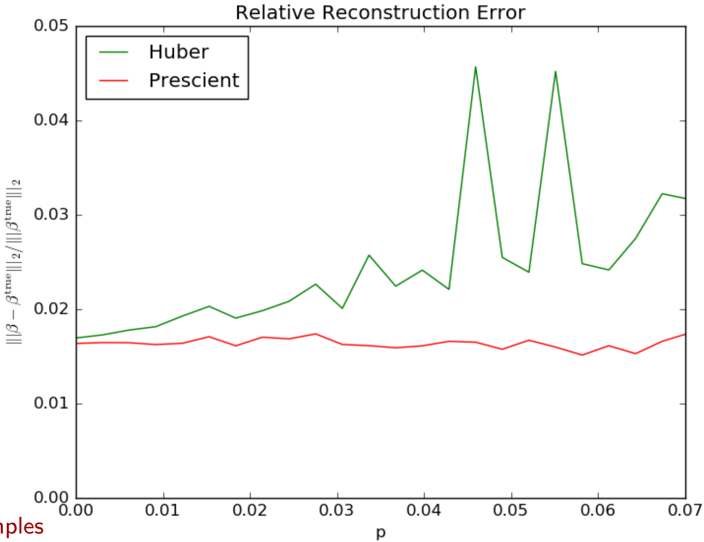
```
beta = Variable(n)
cost = SumEntries(Huber(y - X %*% beta, 1))
prob <- Problem(Minimize(cost))
solution <- solve(prob)
solution$opt_val
solution$beta
```

- Generate data and replace fraction p of y_i's with $-y_i$
- Huber regression with threshold of 1, no constraints
- Compare with OLS and prescient regression where sign changes are known

Results: Huber Regression



Results: Huber Regression



Overview: Direct Standardization

- TODO: Overview of problem

Overview: Direct Standardization

$$\begin{array}{ll} \text{maximize} & \sum_i p_i \log p_i \\ \text{subject to} & p \geq 0, \sum_i^n p_i = 1 \\ & Ap = b \end{array}$$

with variable $p \in \mathbf{R}^n$

- Probabilities p must be non-negative and sum to 1
- $Ap = b$ represents distributional knowledge about some attributes, e.g. expected value, CDF in given range, etc
- Maximizing negative entropy without constraint $Ap = b$ yields uniform distribution

Example: Direct Standardization

- TODO: Specific data we generate for example

Example: Direct Standardization

```
probs <- Variable(n)
cost <- SumEntries(Entr(probs))
constr <- list(probs >= 0, SumEntries(probs) == 1,
   t(X[,1]) %*% probs == 0.5)
prob <- Problem(Maximize(cost), constr)
solution <- solve(prob)
solution$probs
```

- TODO: Details about code, constraint on fraction of women

Results: Direct Standardization

TODO: Plot of CDF from original, skewed, and estimated distributions

Outline

Convex Optimization

CVXR

Code Examples

Disciplined Convex Programming

Future Work

Disciplined Convex Programming (DCP)

(*Grant, Boyd, Ye, 2006*)

- Framework for describing convex optimization problems
- Based on constructive convex analysis
- Sufficient, but not necessary for convexity
- Basis for several domain-specific languages and tools for convex optimization
    - CVX, YALMIP, CVXPY, Convex.jl

Outline

Future Work

- More solvers - SCS, POGS, etc
- More convex functions and constraints
- Domain-specific language for defining new operators
- Warm start to speed up convergence