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, including machine learning and statistics

Convex Problems in Statistics

- Least squares, nonnegative least squares
- Ridge and lasso regression
- Isotonic regression
- Huber (robust) regression
- Logistic regression
- Support vector machine
- Sparse inverse covariance
- Maximum entropy and related problems
- ...and new methods being invented every year!

Domain Specific Languages for Convex Optimization

- Special languages/packages for general convex optimization
- CVX, CVXPY, YALMIP, Convex.jl
- Slower than custom code, but extremely flexible and enables fast prototyping

Domain Specific Languages for Convex Optimization

- Special languages/packages for general convex optimization
- CVX, CVXPY, YALMIP, Convex.jl
- Slower than custom code, but extremely flexible and enables fast prototyping

```
from cvxpy import *
beta = Variable(n)
cost = norm(X * beta - y)
prob = Problem(Minimize(cost))
prob.solve()
beta.value
```

Outline

Convex Optimization

CVXR

Examples

Future Work

CVXR

A modeling language in R for convex optimization

- Will connect to many open source solvers
- Uses disciplined convex programming to verify convexity
- Supports parameters, multiple constraints
- Mixes easily with general R code and other libraries

Outline

Ordinary Least Squares (OLS)

- minimize $||X\beta - y||_2^2$
- $\beta \in \mathbf{R}^m$ is variable, X and y are constants

Ordinary Least Squares (OLS)

- minimize $||X\beta - y||_2^2$
- $\beta \in \mathbf{R}^m$ is variable, X and y are constants

```
library(cvxr)
beta <- Variable(m)
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 returns a list that includes optimal `beta`

Non-Negative Least Squares (NNLS)

- minimize $||X\beta - y||_2^2$ subject to $\beta \geq 0$

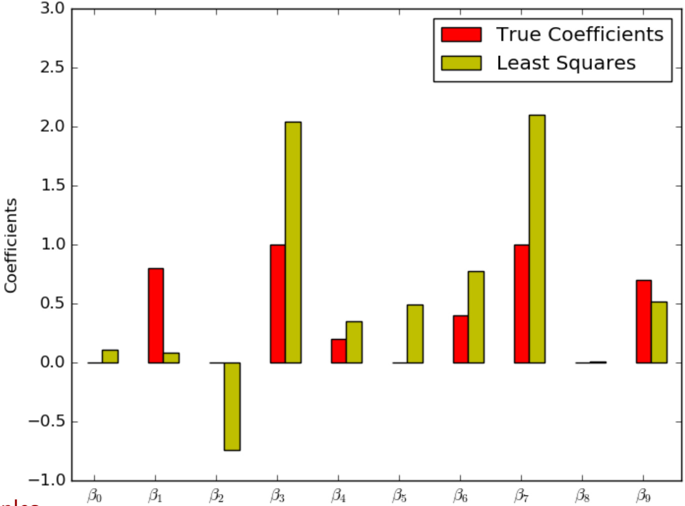
Non-Negative Least Squares (NNLS)

- minimize $||X\beta - y||_2^2$ subject to $\beta \geq 0$

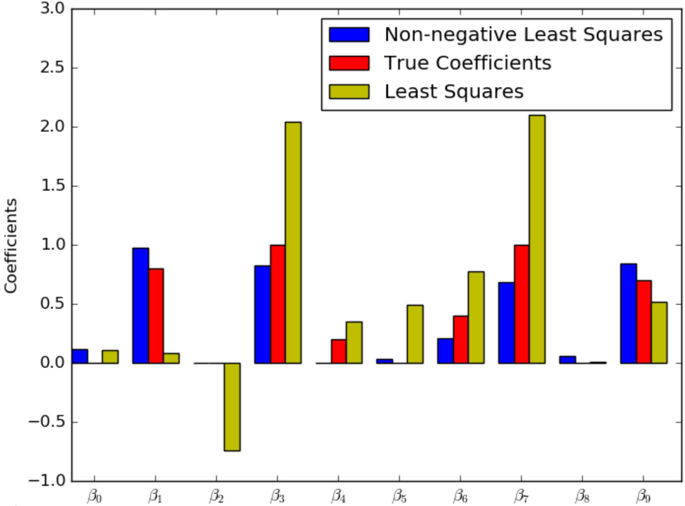
```
constr <- list(beta >= 0)
prob2 <- Problem(Minimize(obj), constr)
solution2 <- solve(prob2)
solution2$opt_val
solution2$beta
```

- Construct new problem with list `constr` of constraints formed from constants and variables
- Variables, parameters, expressions, and constraints exist outside of any problem

Example: True Coefficients and OLS Estimate



True Coefficients and NNLS Estimate



Direct Standardization

- Samples (X, y) drawn **non-uniformly** from a distribution
- Expectations of features of X are known $b \in \mathbf{R}^m$
- We'll estimate probability $p \in \mathbf{R}^n$ for all samples
- Choose p_i to match known expectations, maximize entropy

Direct Standardization

- Samples (X, y) drawn **non-uniformly** from a distribution
- Expectations of features of X are known $b \in \mathbf{R}^m$
- We'll estimate probability $p \in \mathbf{R}^n$ for all samples
- Choose p_i to match known expectations, maximize entropy

$$\begin{array}{ll} \text{maximize} & \sum_i^n \text{entr}(p_i) \\ \text{subject to} & p \geq 0 \quad \mathbf{1}^T p = 1 \quad X^T p = b \end{array}$$

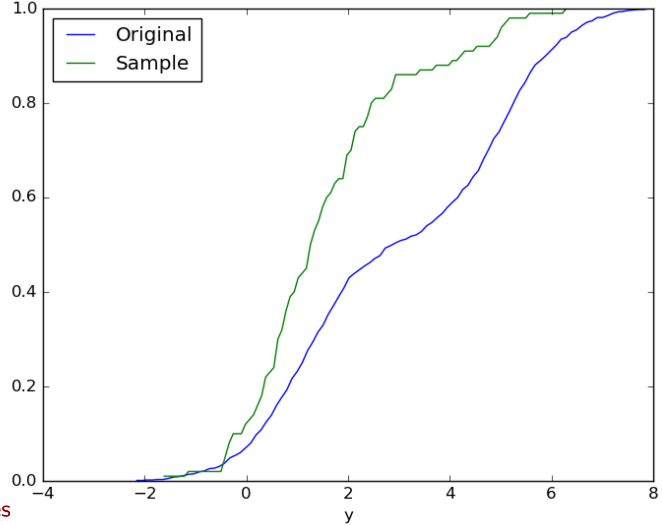
- $\text{entr}(p_i) = -p_i \log p_i$
- (y, p) is an estimate of the true sampling distribution of the response variable

Direct Standardization

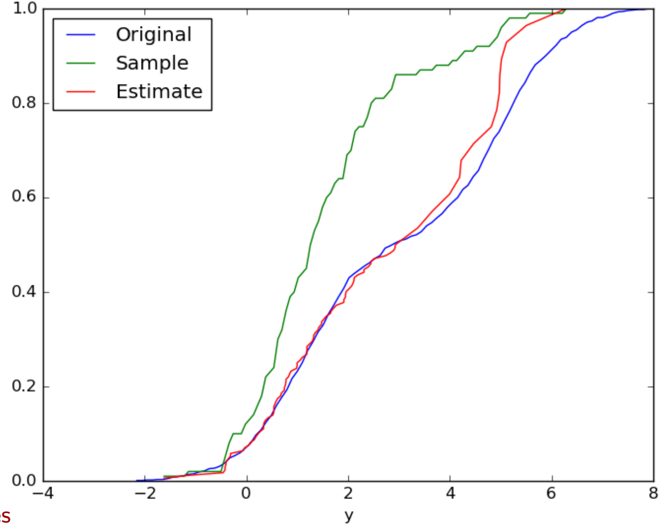
```
probs <- Variable(n)
cost <- SumEntries(Entr(probs))
constr <- list(probs >= 0, SumEntries(probs) == 1,
   t(X) %*% probs == b)
prob <- Problem(Maximize(cost), constr)
solution <- solve(prob)
solution$probs
```

- `Entr` is the elementwise entropy function
- `solution$probs` is an R vector of sample probabilities

True vs. Sample Cumulative Distribution



True vs. Estimated Cumulative Distribution



Outline

Future Work

- Connect to more solvers: SCS, CVXOPT, ...
- Flesh out convex functions in library
- Add warm start support