# Disciplined Convex Optimization with CVXR

**Anqi Fu** Bala Narasimhan Stephen Boyd

EE & Statistics Departments

Stanford University

useR! Conference 2018

# 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

# CVXR

A modeling language in R for convex optimization

- Connects to many solvers: ECOS, SCS, MOSEK, etc
- Mixes easily with general R code and other libraries
- Uses disciplined convex programming to verify convexity

# Outline

## Ordinary Least Squares (OLS)

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

## Ordinary Least Squares (OLS)

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

```
library(CVXR)
beta <- Variable(n)
obj <- sum_squares(y - X %*% beta)
prob <- Problem(Minimize(obj))
result <- solve(prob)
result$value
result$getValue(beta)
```

- `X` and `y` are constants; `beta`, `obj`, and `prob` are S4 objects
- `solve` method returns a list that includes optimal `beta` and objective `value`

## Non-Negative Least Squares (NNLS)

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

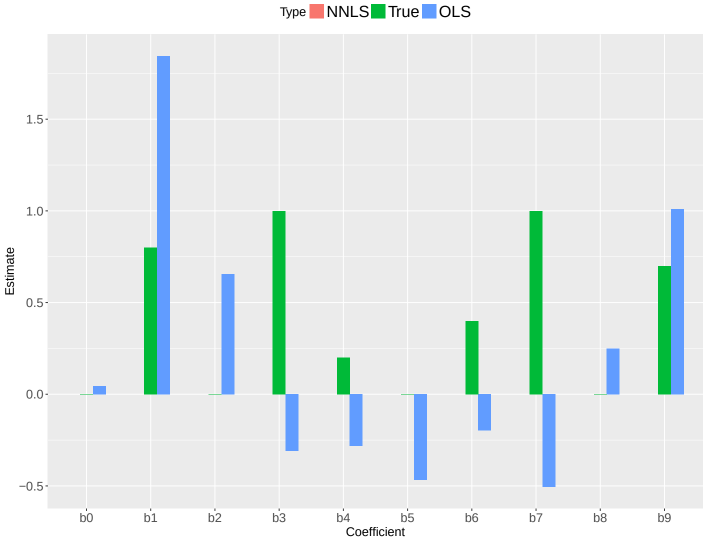
## Non-Negative Least Squares (NNLS)

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

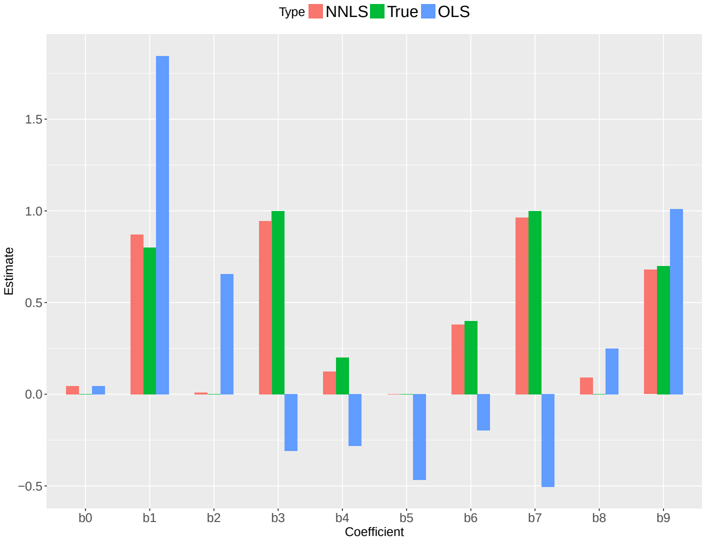
```
constr <- list(beta >= 0)
prob2 <- Problem(Minimize(obj), constr)
result2 <- solve(prob2)
result2$value
result2$getValue(beta)
```

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

# True vs. Estimated Coefficients



# True vs. Estimated Coefficients



## Sparse Inverse Covariance Estimation

- Samples $x_i \in \mathbf{R}^n$ drawn i.i.d. from $N(0, \Sigma)$
- Know covariance $\Sigma \in \mathbf{S}_+^n$ has **sparse** inverse $S = \Sigma^{-1}$

## Sparse Inverse Covariance Estimation

- Samples $x_i \in \mathbf{R}^n$ drawn i.i.d. from $N(0, \Sigma)$
- Know covariance $\Sigma \in \mathbf{S}^n_+$ has **sparse** inverse $S = \Sigma^{-1}$
- One way to estimate $S$ is by maximizing the log-likelihood with a sparsity constraint:

$$\begin{array}{ll} \underset{S}{\text{maximize}} & \log\det(S) - \mathrm{tr}(SQ) \\ \text{subject to} & S \in \mathbf{S}^n_+, \quad \sum_{i=1}^n \sum_{j=1}^n |S_{ij}| \leq \alpha \end{array}$$

- $Q = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})(x_i - \bar{x})^\top$ is sample covariance
- $\alpha \geq 0$ is a parameter controlling the degree of sparsity

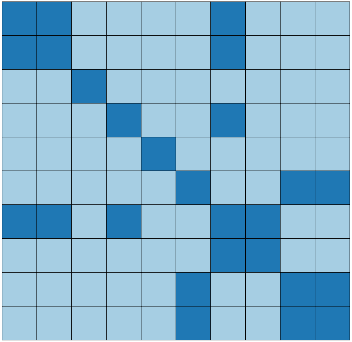
## Sparse Inverse Covariance Estimation

```
S <- Semidef(n)
obj <- log_det(S) - matrix_trace(S %*% Q)
constr <- list(sum(abs(S)) <= alpha)
prob <- Problem(Maximize(obj), constr)
result <- solve(prob)
result$getValue(S)
```

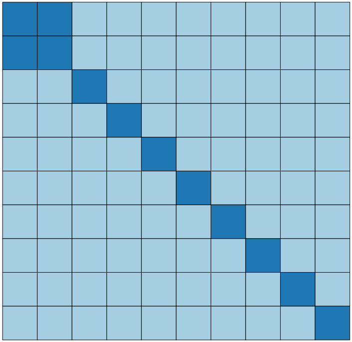
- `Semidef` restricts variable to positive semidefinite cone
- Must use `log_det(S)` instead of `log(det(S))` since `det` is not a supported atom
- `result$getValue(S)` returns an R matrix

# True vs. Estimated Sparsity of Inverse

True Inverse

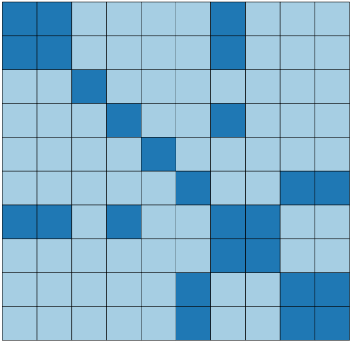


Estimate $(\alpha = 1)$

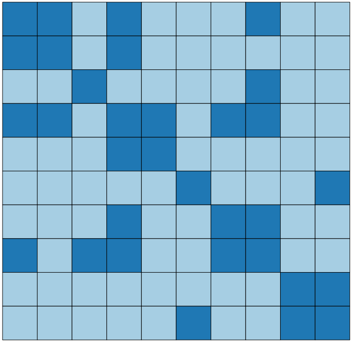


# True vs. Estimated Sparsity of Inverse

True Inverse

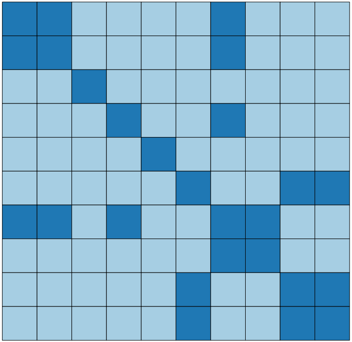


Estimate ($\alpha = 6$)

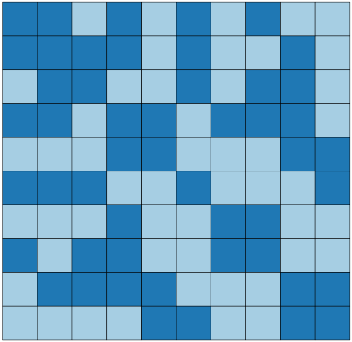


# True vs. Estimated Sparsity of Inverse

True Inverse



Estimate ($\alpha = 10$)



## Outline

Convex Optimization

CVXR

Examples

Future Work

## Future Work

- Flesh out convex functions in library
- Develop more applications and examples
- Make connecting new solvers easier
- Add warm start support
- Further speed improvements

Official site: `cvxr.rbind.io`
CRAN page: `CRAN.R-project.org/package=CVXR`