The CVXR Package

Hans W Borchers, Duale Hochschule Mannheim

10/04/2018

What is "Convex Optimization" ?

"**Convex minimization** is a subfield of optimization that studies the problem of minimizing convex functions over convex sets."
–Wikipedia

- Very fast algorithms (like Linear Programming, LP)
- Convex problems have only *one* (global) optimum
- Many statistical and engineering applications can be modeled as convex problems
- BUT: May be difficult to find an appropriate convex formulation (NP-hard)

Convex Functions and Domains

A function $f : \mathbf{R}^n \to \mathbf{R}$ is *convex* if its domain of definition is convex and for all x, y and $0 \leq \theta \leq 1$ we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

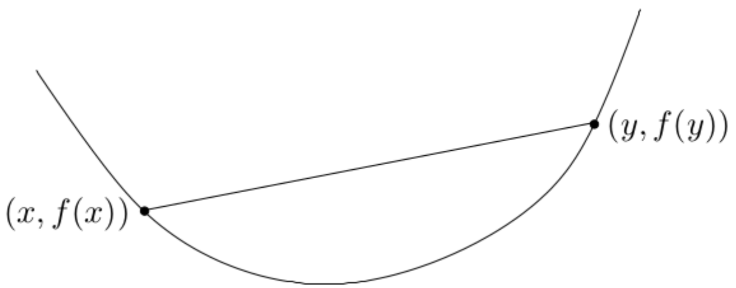


Figure 1: Figure: Graph of a convex function (Boyd et al. 2004)

What is CVX* ?

CVX* is a family of implementations of **Disciplined Convex Programming** (DCP), invented an initialized by *Stephen Boyd* and collaborators at Stanford University:

- CVX (MATLAB, ~2005)
- CVXPY (Python, 2013)
- convex.jl (Julia, 2015)
- CVXR (R, 2017)

Disciplined convex programming imposes a set of conventions to follow when constructing convex problems.

Loading CVXR

```
devtools::install_github("anqif/CVXR")
vignette("cvxr_intro", package="CVXR")
```

```
# suppressMessages(suppressWarnings(library(CVXR)))
library(CVXR)
```

```
##
## Attaching package: 'CVXR'

## The following object is masked from 'package:stats':
##
##     power
```

```
package?CVXR
```

Linear Regression with CVXR

```
x <- Variable(11)
objective  <- Minimize(sum((b - A %*% x)^2))
problem    <- Problem(objective)
result     <- solve(problem)
c( result$getValue(x) )
```

```
##  [1] -0.05059088 -1.95850906 -0.02934818  0.02498838 -0
##  [6]  0.00479077 -0.00087763  2.04205369  0.16839344  0
## [11]  0.36563333
```

Example: Linear Regression

```
wine <- read.csv("winequality.csv", sep=";")

mod0 <- lm(quality ~ . - 1, data=wine)
unname(coefficients(mod0))
```

```
##  [1] -0.05059062 -1.95851023 -0.02934924  0.02498840 -0.
##  [6]  0.00479079 -0.00087763  2.04204607  0.16839514  0.
## [11]  0.36563338
```

```
A <- wine[, 1:11]; b = wine[, 12]
mod00 <- qr.solve(A, b)
unname(mod00)
```

```
##  [1] -0.05059062 -1.95851023 -0.02934924  0.02498840 -0.
##  [6]  0.00479079 -0.00087763  2.04204607  0.16839514  0.
## [11]  0.36563338
```

Positive Coefficients only

```
x <- Variable(11)
objective   <- Minimize(sum((b - A %*% x)^2))
constraint <- list(x >= 0)
problem     <- Problem(objective, constraint)
result      <- solve(problem)
c( result$getValue(x) )
```

```
##  [1] -1.0927e-10  4.6149e-11  1.1556e-01  1.9532e-02  4.
##  [6]  5.0505e-03 -3.0693e-10  4.4630e-01  3.3277e-01  3.
## [11]  3.6525e-01
```

A 'Sum Equal to 1' Solution

```
x <- Variable(11)
objective  <- Minimize(sum((b - A %*% x)^2))
constraint <- list(x >= 0, sum(x) == 1)
problem    <- Problem(objective, constraint)
result     <- solve(problem)
zapsmall( c( result$getValue(x) ) )
```

```
##  [1] 0.00000 0.00000 0.00000 0.02209 0.00000 0.00554 0.(
##  [9] 0.46537 0.12725 0.37975
```

```
sum(result$getValue(x))
```

```
## [1] 1
```

L1 Regression

"L1 regression, or Least Absolute Deviations (LAD) regression, is a statistical optimality criterion and the statistical optimization technique that relies on minimizing the L1-norm."

Linear L1 regression: $\text{Min! } \sum_1^n |\, b - A\,x|$

```
x <- Variable(11)
objective  <- Minimize(sum(abs(b - A %*% x)))
constraint <- list(x[11] == 0)
problem    <- Problem(objective, constraint)
result     <- solve(problem)
c( result$getValue(x) )
```

```
##  [1] -1.0958e-02 -4.6122e-01  1.9429e-02 -2.0589e-03 -5
##  [6]  1.3345e-03 -7.1307e-04  6.2513e+00  5.5697e-02  1
## [11] -1.2074e-11
```

'Isotonic' Regression

"In statistics, isotonic regression or monotonic regression is the technique of fitting a free-form line to a sequence of observations under the [monotone] constraints." – Wikipedia

Example: `x[1]<=x[2]<=...<=x[n]`

```
x <- Variable(11)
objective  <- Minimize(sum((b - A %*% x)^2))
constraint <- list(diff(x) >= 0)
problem    <- Problem(objective, constraint)
result     <- solve(problem)
c( result$getValue(x) )
```

```
##  [1] -0.0212915 -0.0212915  0.0016911  0.0016911  0.001(
##  [7]  0.0016911  0.3767073  0.3767073  0.3767073  0.376
```

Robust Regression

"Robust regression is a form of regression analysis designed to overcome some limitations of traditional parametric and non-parametric methods, especially high sensitivity to outliers."

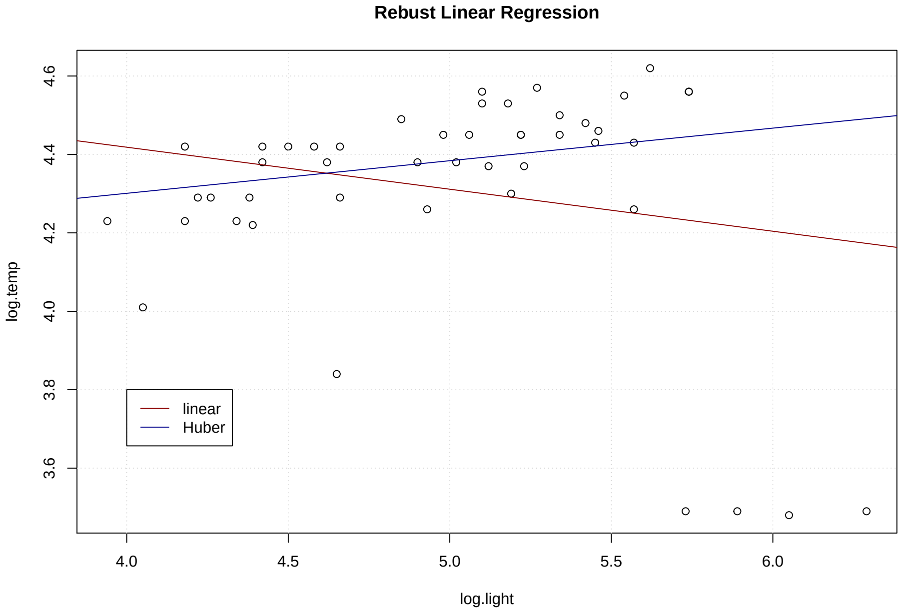
Huber's M-estimation: $\text{Min! } \sum L_M(b - A\,x)$ with $L_M(u) = \frac{1}{2}u^2$ if $|u| \leq M$, else $2M|u| - M^2$.

```
M <- 1   # Huber threshold
x <- Variable(11)
objective  <- Minimize(sum(huber(b - A %*% x, M)))
problem    <- Problem(objective)
result     <- solve(problem)
c( result$getValue(x) )
```

```
##  [1] -0.0478472 -1.8331217  0.0153206  0.0216320 -0.936(
##  [7] -0.0011800  1.8376328  0.2056466  0.4656358  0.366
```

Example: Robust Regression

Stars outer temperature vs. light intensity:



Example continued . . .

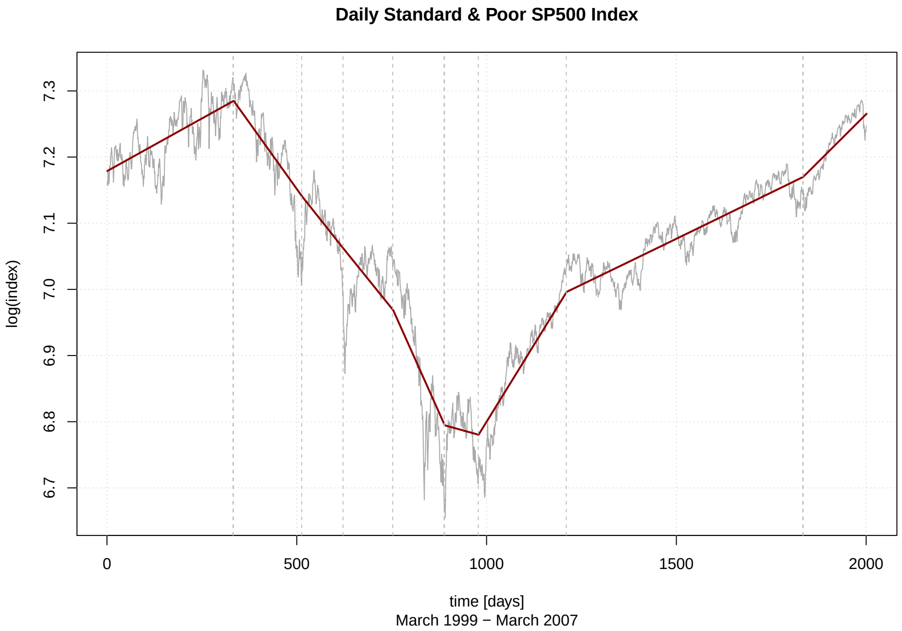
```
# library(CVXR)

stars = read.csv("starscyg.csv")
A = cbind(1, stars$log.light)
b = stars$log.temp

M <- 0.2  # Huber threshold
x <- Variable(2)
objective  <- sum(huber(b - A %*% x, M))
problem    <- Problem(Minimize(objective))
result     <- solve(problem)
ab = result$getValue(x)
ab
```

```
##           [,1]
## [1,] 3.968469
## [2,] 0.083105
```

Example: Piecewise Linear Regression



Example Solved with CVXR

One approach to ‘piecewise linear regression’ is through this formula:

$$\text{Min!}\frac{1}{2}\sum_{1}^{n}(y_i - z_i)^2 + \lambda \sum_{1}^{n-2} |z_i - 2z_{i+1} + z_{i+2}|$$

```
lambda = 40
z <- Variable(length(y))
objective <- 0.5 * p_norm(y - z) +
                lambda * p_norm(diff(z, differences = 2), 1
problem  <- Problem(Minimize(objective))
sol <- solve(problem)$getValue(z)
```

Quadratic Optimization

Quadratic Programming (QP) is the problem of optimizing a quadratic expression of several variables subject to linear constraints.

$$\text{Minimize} \quad \frac{1}{2}x^TQx + c^Tx \qquad s.t. \quad Ax \leq b$$

where
Q is a symmetric, positive (semi-)definite $n \times n$-matrix,
c an n-dim. vector,
A an $m \times n$-matrix, and
b an m-dim. vector.

For some solvers, linear equality constraints are also allowed.

Quadratic Optimization CRAN Optimization Task

Example (continued)

As an example, we will look at finding a smallest circle enclosing 100 randomly given points $p_1, \ldots, p_{100}$ in $\mathbb{R}^2$. We will represent the coordinates of these points as columns in the following matrix `P`.

```
set.seed(7531); N <-  100
P <- matrix(10*rnorm(2*N), nrow=2)
# plot(P[1, ], P[2, ], col="red", xlab="", ylab="")
```

```
C <- t(P) %*% P
d <- apply(P^2, 2, sum)
```

Example: Smallest Enclosing Ball

Given a set $P = \{p_1, ..., p_n\}$ of n points in $\mathbb{R}^d$, find a point p_0 such that $max||p_i - p_0||$ is minimized.

Known algorithm to solve this as Qudratic Programming task:

Define matrix $C = (p_1, ..., p_n)$, i.e. coordinates of points in columns, and minimize the quadratic form

$$x^TC^TCx - \sum p_i^Tp_ix_i$$

subject to $\sum x_1 = 1$ and all $x_i >= 0$.

Let $x = (x_1, ..., x_n)$ be an optimal solution, then the linear combination $p0 = \sum x_ip_i$ is the center of the smallest enclosing ball, and the negative of the minimum value at x is the square of the radius of the ball.

Example Solved with CVXR

```
x            <- Variable(N)
objective    <- Minimize( quad_form(x, C) - sum(d * x))
constraint   <- list(x >= 0, sum(x) == 1)
problem      <- Problem(objective, constraint)
result       <- solve(problem, solver="SCS")      # default:
```

```
x0 <- result$getValue(x)
p0 <- P %*% x0; c(p0)
```

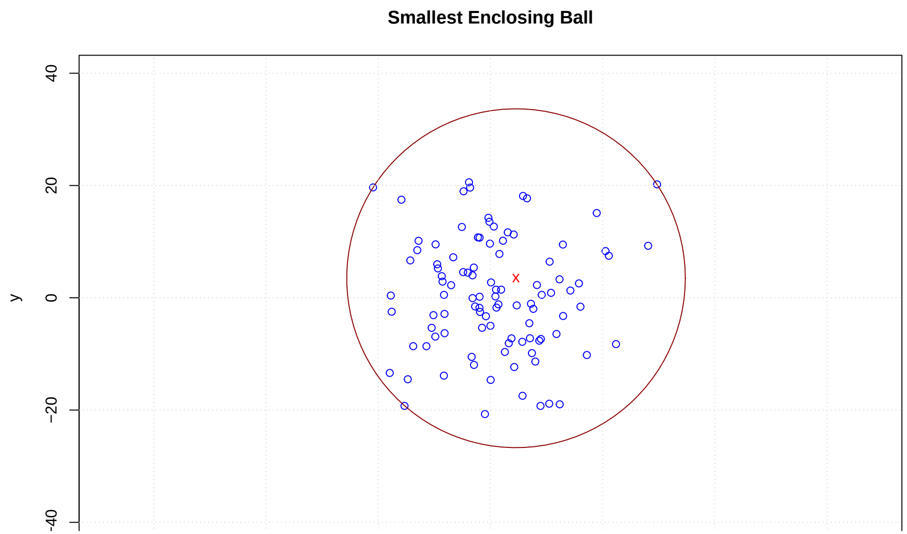
```
## [1] 4.5508 3.4853
```

```
r0 <- c(sqrt(sum(colSums(P^2)*x0) - t(x0)%*%t(P)%*%P%*%x0))
r0
```

```
## [1] 30.182
```

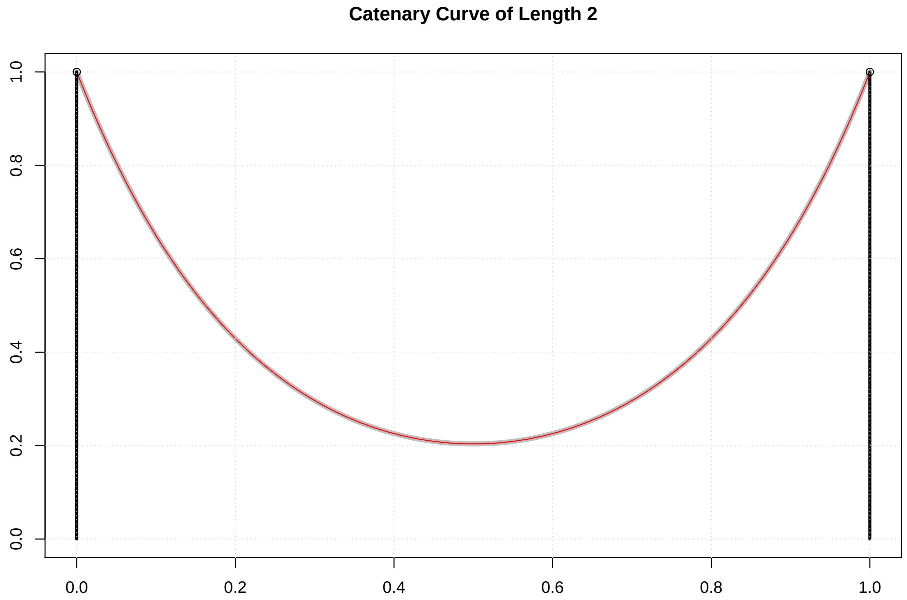
Example Solution

```r
plot(P[1, ], P[2, ], xlim=c(-40,40), ylim=c(-40,40), asp=1,
     col = "blue", xlab = "x", ylab = "y",
     main = "Smallest Enclosing Ball")
# ...
```



Example: Catenary

Solve the "hanging chain curve" as an optimization problem!
See hwborchers.lima-city.de/Presents/catenary.html.



CVXR Tutorial Examples

- Largest Euclidean ball in a 2D polyhedron
- Catenary Problem
- Huber Regression
- Logistic Regression
- Quantile Regression
- Censored Regression
- Isotonic Regression
- Near Isotonic and Near Convex Regression
- L1 Trend Filtering
- Elastic Net
- Saturating Hinges
- Direct Standardization
- Log-Concave Density Estimation
- Sparse Inverse Covariance Estimation
- Kelly Gambling
- Fastest Mixing Markov Chain
- Portfolio Optimization

Catenary Solved with CVXR

```r
N <- 100; L <- 2
h <- L / (N-1)
x <- Variable(N)
y <- Variable(N)
objective  <- Minimize(sum(y))
constraint <- list(x[1]==0, x[N]==1, y[1]==1, y[N]==1,
                   diff(x)^2 + diff(y)^2 <= h^2)
problem <- Problem(objective, constraint)
result <- solve(problem)    ## solver="SCS"
xm <- result$getValue(x)
ym <- result$getValue(y)
#  result
## $status:     "optimal"
## $solver:     "ECOS"
## $solve_time: 0.008145835
## $setup_time: 0.000476103
```

Running Times – Catenary Example

Solver	N = 50	N = 100	N = 1000
auglag	8.0	60 [–]	–
Ipopt			
CVXR/ECOS	0.283	0.297	NA
CVXR/SCS	0.311	0.330 [-]	1.141 [–]
ECOS	0.002	0.003	0.036
SCS	0.002	0.010	0.280
Rmosek	0.004	0.005	0.033
JuMP	0.007	0.016	0.416

Web Links

- See anqif/CVXR on Github
- CVXR Package vignette
- CVXR Home page
- CVXR Tutorial examples
- CVXR Function reference
- Anqi Fu's talk Disciplined Convex Optimization with CVXR at UseR!2016, Stanford University
- A. Fu, , B. Narasimhan, and Stephen Boyd. CVXR: An R Package for Disciplined Convex Optimization, Manuscript Draft, 2018.

Reference

A. Fu, B. Narasimhan, and S. Boyd (2018).*CVXR: An R Package for disciplined convex optimization*. Journal of Statistical Software. [To be published.]

Acknowledgements

The authors would like to thank Trevor Hastie, Robert Tibshirani, John Chambers, David Donoho, and Hans Werner Borchers for their thoughtful advice and comments on this project. We are grateful to Steven Diamond, John Miller, and Paul Kunsberg Rosenfield for their contributions to the software's development. In particular, we are indebted to Steven for his work on CVXPY. Most of CVXR's code, documentation, and examples were ported from his Python library