In-class session 3

Ishwara Hegde

01/01/2021

The goal of this in-class session was :

1. Understanding balancing problems as a convex optimization problem.
2. Solve the convex optimization problem using CVXR
3. Identify OLS as simply a convex opt problem and identify why this lends well to the BLUE representation of OLS.

Preamble

Loading packages and setting seed :

```
#Loading the packages needed:

#install.packages('CVXR')

library(mvtnorm)
library(CVXR)


#Setting the seed and clearing workspace
set.seed(1234)
rm(list = ls())
```

This is Dmitry's code that he uses for plotting densities. I prefer to use my own plotting function in ggplot that you can see in later assignmnets.

```
my_density_function <- function(x,K,deg){

    x_min <- min(x)
    x_max <- max(x)
    range_x <- x_max - x_min
    low_x <- x_min - 0.2*range_x
    up_x <- x_max + 0.2*range_x
    range_full <-up_x- low_x
    splits <- seq(low_x,up_x,length.out = K+1)
    mesh_size <- splits[2] - splits[1]
    centers <- (splits[-1]+splits[-(K+1)])/2
    counts <- as.vector(table(cut(x,splits,include.lowest = TRUE)))
    scale <- sum(counts)*mesh_size

    data_matrix <- splines::ns(centers, df = deg)
    pois_reg_res <- glm(counts~data_matrix, family = 'poisson')
    freq_pois <- exp(pois_reg_res$linear.predictors)
    dens_pois <- freq_pois/scale
```

```r
    return(cbind(centers,freq_pois,dens_pois))
}
```

Just setting up some basic parameters for the problem:

```r
p <- 10
n <- 100
beta_0 <- (1:p)^{-2}
beta_0_norm <- beta_0/sqrt(sum(beta_0^2))
gamma <- beta_0_norm

# tau is the treatment effect
tau_av <- 1
sigma_0 <- sqrt(0.2)
sigma_1 <- sqrt(0.2)


X <-  rmvnorm(n,sigma = diag(rep(1,p)))
logit <- X%*%gamma*2
pi <- exp(logit)/(1+exp(logit))
pi_tau <- as.numeric(pi > 0.5)
tau_het <- 1 + (2*pi_tau-1)*1


noise_0 <- rnorm(n,sd = sigma_0)
noise_1 <- rnorm(n,sd = sigma_1)
W <- rbinom(n,1,pi)
Y_0 <- X%*%beta_0_norm + noise_0
Y_1 <- tau_het + X%*%beta_0_norm + noise_1
tau_cond <- mean(tau_het)
Y <- Y_0*(1-W) + Y_1*W
```

Problem 1

Estimate tau using a linear regression of Y on W and X, and using CVXR to solve the balancing problem.

```r
#See the ols results below
tau_ols <- lm(Y~W + X)$coefficients[2]
tau_ols
```

```
##        W
## 1.138206
```

```r
#The OLS is biased when there is heterogeneity in tau

w_unit <- Variable(n,name = 'weights')
obj <- Minimize(sum(w_unit^2)/(n^2))
constr <- list(
    t(w_unit)%*%W/n ==1,
    t(w_unit)%*%X/n ==0,
    sum(w_unit)/n == 0



)
```

```r
problem <- Problem(obj, constraints  = constr)
result <- psolve(problem)
weights_init <- result[[1]]

tau_bal <- t(weights_init)%*%Y/n


cbind(round(weights_init,3),W)
```

```
##              W
##   [1,] -2.264 0
##   [2,] -1.201 0
##   [3,]  2.807 1
##   [4,]  1.676 1
##   [5,]  0.687 1
##   [6,] -0.889 0
##   [7,]  3.267 1
##   [8,]  3.073 1
##   [9,]  4.368 1
##  [10,] -2.715 0
##  [11,] -4.810 0
##  [12,]  5.914 1
##  [13,]  3.510 1
##  [14,]  1.456 1
##  [15,] -4.078 0
##  [16,] -3.326 0
##  [17,] -0.796 0
##  [18,] -0.742 1
##  [19,]  2.540 0
##  [20,]  3.826 1
##  [21,]  2.602 1
##  [22,]  3.301 1
##  [23,]  0.933 1
##  [24,] -2.921 0
##  [25,]  4.372 1
##  [26,] -5.616 0
##  [27,]  0.702 1
##  [28,]  3.643 1
##  [29,]  0.489 0
##  [30,] -5.980 0
##  [31,] -1.319 0
##  [32,]  1.960 1
##  [33,]  1.832 1
##  [34,]  3.214 1
##  [35,] -4.237 0
##  [36,]  4.044 0
##  [37,]  4.544 1
##  [38,] -2.573 0
##  [39,]  2.747 0
##  [40,]  0.863 1
##  [41,] -1.493 0
##  [42,] -1.395 0
##  [43,] -0.262 0
```

```
## [44,]  1.457 0
## [45,] -3.119 0
## [46,]  2.884 1
## [47,]  4.550 1
## [48,] -0.349 1
## [49,]  1.067 0
## [50,]  2.245 1
## [51,]  0.697 1
## [52,] -0.168 0
## [53,]  0.145 1
## [54,] -4.683 0
## [55,]  3.320 1
## [56,] -0.673 0
## [57,] -2.714 0
## [58,] -4.711 0
## [59,] -1.996 0
## [60,] -4.884 0
## [61,] -2.834 0
## [62,] -2.377 0
## [63,]  1.742 1
## [64,]  1.826 1
## [65,]  3.312 1
## [66,] -1.826 0
## [67,] -2.342 0
## [68,]  1.219 1
## [69,]  3.146 1
## [70,] -4.947 0
## [71,]  1.115 1
## [72,]  0.717 1
## [73,] -0.118 0
## [74,]  0.301 1
## [75,]  2.559 1
## [76,]  3.866 1
## [77,] -3.729 0
## [78,] -2.431 1
## [79,]  0.038 1
## [80,]  2.795 1
## [81,] -0.793 0
## [82,]  0.497 1
## [83,] -0.465 0
## [84,] -3.021 0
## [85,] -2.517 0
## [86,] -2.110 0
## [87,] -0.470 0
## [88,] -3.117 0
## [89,] -2.456 0
## [90,]  6.642 1
## [91,]  4.641 0
## [92,] -5.249 0
## [93,] -1.689 0
## [94,] -4.927 0
## [95,] -0.263 1
## [96,] -1.571 0
## [97,] -0.533 1
```

```
##  [98,] -1.605 0
##  [99,] -1.558 1
## [100,]  3.709 1
```

Notice that some units get negative weights. Suppose we do not want this, i.e. we want to remain in the convex hull of treatment effects?

Problem 2

Estimate tau using CVXR imposing the non-negativeity constraints

```
w_unit <- Variable(n,name = 'weights')
obj <- Minimize(sum(w_unit^2)/(n^2))
constr <- list(
    t(w_unit)%*%W/n ==1,
    t(w_unit)%*%X/n ==0,
    sum(w_unit)/n == 0,
    W*w_unit >=0,
    (1-W)*w_unit <=0

)
problem <- Problem(obj, constraints  = constr)
result <- psolve(problem)
weights_conv <- result[[1]]

tau_bal_int <- t(weights_conv)%*%Y/n
```

Now we notice tau is closer to the actual value of 1 as compared to when we used the unrestricted weights.

Problem 3

Repeat exericise 2 for B = 500 simulations (simulate only W and Y, fixing X) and plot the distribution of the estimators

```
B <- 500
results <- matrix(0, ncol = 2, nrow = B)
for (b in 1:B){


    W_b <- rbinom(n,1,pi)
    noise_0_b <- rnorm(n,sd = sigma_0)
    noise_1_b <- rnorm(n,sd = sigma_1)
    Y_0_b <- X%*%beta_0_norm + noise_0_b
    Y_1_b <- tau_het + X%*%beta_0_norm + noise_1_b


    Y_b <- Y_0_b*(1-W_b) + Y_1_b*W_b

    w_unit_b <- Variable(n,name = 'weights')
    obj_b <- Minimize(sum(w_unit_b^2)/(n^2))
    constr_b <- list(
        t(w_unit_b)%*%W_b/n ==1,
        t(w_unit_b)%*%X/n ==0,
        sum(w_unit_b)/n == 0,
        W_b*w_unit_b >=0,
        (1-W_b)*w_unit_b <=0
```

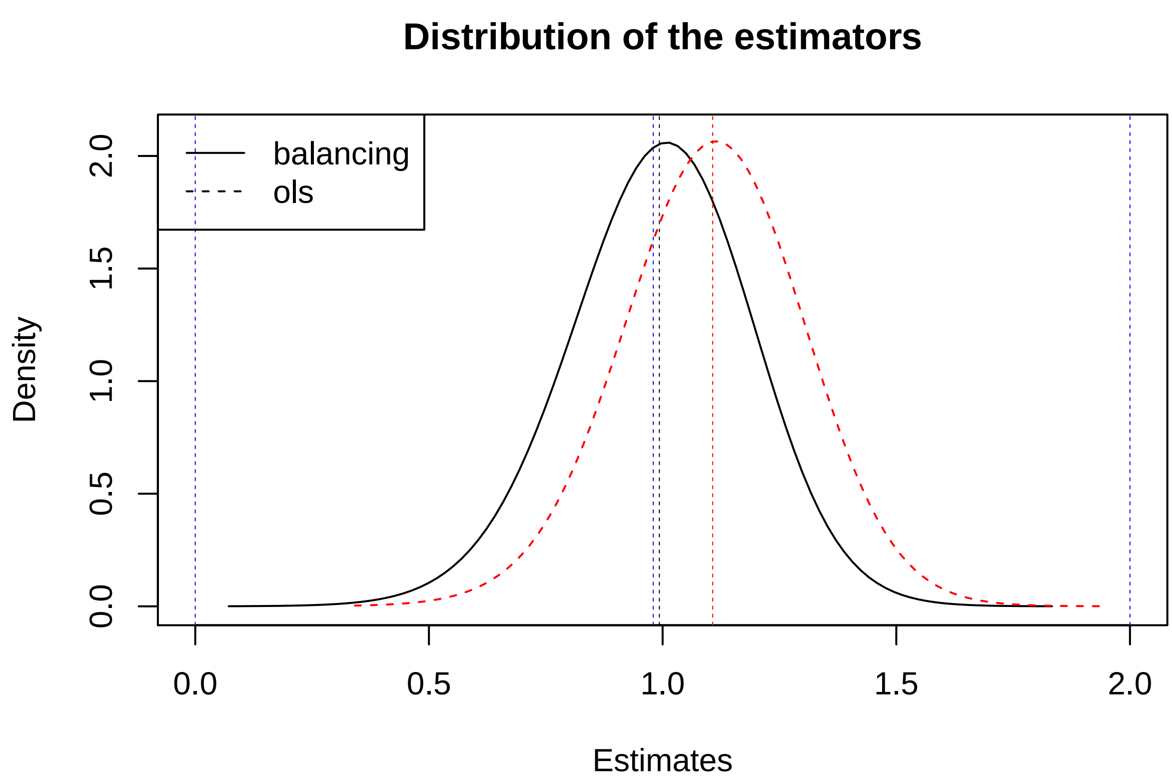
```
    )
    problem <- Problem(obj_b, constraints  = constr_b)
    result_b <- psolve(problem)
    weights_b <- result_b[[1]]
    tau_bal_int_b <- as.vector(t(weights_b)%*%Y_b/n)

    tau_ols_b <- lm(Y_b~W_b+X)$coefficients[2]

    results[b,] <- c(tau_bal_int_b, tau_ols_b)
    #print(b)
}

bal_est <- my_density_function(results[,1],100,deg = 3)[,c(1,3)]
ols_est <- my_density_function(results[,2],100,deg = 3)[,c(1,3)]


plot(bal_est,main = 'Distribution of the estimators', col = 'black',lty = 1,ylim = c(0,2.1),xlim = c(0,
lines(ols_est, col = 'red',lty = 2)
abline(v = mean(results[,1]), lty = 2,lwd = 0.5,col = 'black')
abline(v = mean(results[,2]), lty = 2,lwd = 0.5,col = 'red')
abline(v =0, lty = 2,lwd = 0.5,col = 'blue')
abline(v = 2, lty = 2,lwd = 0.5,col = 'blue')
abline(v =tau_cond, lty = 2,lwd = 0.5,col = 'blue')
legend('topleft',col <- c('black','red',), lty = c(1,2),
legend = c('balancing','ols'))
```



The red is the OLS estimate and the black is the balancing estimate with a nonnegative weight constraint. Notice how the balancing estimator is not the same as the OLS estimate anymore.