In-class session 5

Ishwara Hegde

01/01/2021

The goal of this in-class session was :

1. Understand the implementation of cross-fitting and see how it differs from cross validation.
2. Implement a doubly robust estimator for tau (TE parameter) and compare it to a simple difference in means that is obtained by cross fitting!

Preamble

Loading packages and setting seed :

```
#Loading the packages needed:

#install.packages('CVXR')
library(CVXR)
library(mvtnorm)
library(glmnet)


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

Using Dmitrys plot function. Again, I find it a bit too complicated. An alternative with ggplot is easier and I have implemented it in the homework exercises.

```
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
```

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

Setting up some parameters:

```
p <- 200
n <- 200
beta_0 <- (1:p)^{-2}
beta_0_norm <- beta_0/sqrt(sum(beta_0^2))
gamma <- beta_0_norm
sigma_0 <- sqrt(0.2)
sigma_1 <- sqrt(0.2)
lambda <- 100



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
```

Implementing a double-robust estimator:

```
B <- 200
results <- matrix(0, ncol = 2, nrow = B)

for (b in 1:B){


    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


    W_1 <- as.numeric(W==1)
    W_0 <- as.numeric(W==0)

    index_1 <- sample(c(TRUE,FALSE),n,replace = TRUE)
    index_2 <- !index_1


    data_x_11 <- X[index_1 & W_1 == 1,]
    data_y_11 <- Y[index_1 & W_1 == 1,]


    cv_glm_11 <- cv.glmnet(data_x_11, data_y_11, family = "gaussian")
    lambda_11 <- cv_glm_11$lambda.min
    glm_opt_11 <- glmnet(data_x_11, data_y_11, family = "gaussian",
                         lambda =lambda_11)
```

```
    data_x_10<- X[index_1 & W_0 == 1,]
    data_y_10 <- Y[index_1& W_0 == 1,]

    cv_glm_10 <- cv.glmnet(data_x_10, data_y_10, family = "gaussian")
    lambda_10 <- cv_glm_10$lambda.min
    glm_opt_10 <- glmnet(data_x_10, data_y_10, family = "gaussian",
                         lambda =lambda_11)


    data_x_21 <- X[index_2 & W_1 == 1,]
    data_y_21 <- Y[index_2& W_1 == 1,]

    cv_glm_21 <- cv.glmnet(data_x_21, data_y_21, family = "gaussian")
    lambda_21 <- cv_glm_21$lambda.min
    glm_opt_21 <- glmnet(data_x_21, data_y_21, family = "gaussian",
                         lambda =lambda_11)


    data_x_20<- X[index_2& W_0 == 1,]
    data_y_20 <- Y[index_2& W_0 == 1,]

    cv_glm_20 <- cv.glmnet(data_x_20, data_y_20, family = "gaussian")
    lambda_20 <- cv_glm_20$lambda.min
    glm_opt_20 <- glmnet(data_x_20, data_y_20, family = "gaussian",
                         lambda =lambda_11)


## prediction

    fit_11 <- predict(glm_opt_21,X)
    fit_21 <- predict(glm_opt_11,X)
    fit_10 <- predict(glm_opt_20,X)
    fit_20<- predict(glm_opt_10,X)
    hat_m_1 <- rep(0,n)
    hat_m_0 <- rep(0,n)
    hat_m_1[index_1] <- fit_11[index_1]
    hat_m_1[index_2] <- fit_21[index_2]
    hat_m_0[index_1] <- fit_10[index_1]
    hat_m_0[index_2] <- fit_20[index_2]

# Now finding the weights as a balancing problem:

    w_unit_1 <- Variable(n)
    t_1 <- Variable(1)
    obj <- Minimize(1/n^2*sum((W_1*w_unit_1)^2) + lambda*t_1^2)
    constr <- list(
        abs(t(W_1*w_unit_1-1)%*%X/n) <= t_1,
        sum(w_unit_1*W_1)/n == 1
    )

    problem <- Problem(obj, constraints  = constr)
    result <- psolve(problem)
    weights_upd_1 <- result[[1]]
```

```r
    w_unit_0 <- Variable(n)
    t_0 <- Variable(1)
    obj <- Minimize(1/n^2*sum((W_0*w_unit_0)^2) + lambda*t_0^2)
    constr <- list(
        abs(t(W_0*w_unit_0-1)%*%X/n) <= t_0,
        sum(w_unit_0*W_0)/n == 1
    )

    problem <- Problem(obj, constraints  = constr)
    result <- psolve(problem)
    weights_upd_0 <- result[[1]]


# Now constructing the estimators tau_pred and tau_db

    tau_pred <- mean(hat_m_1 - hat_m_0)
    tau_db <- tau_pred + mean(W_1*(Y-hat_m_1)*weights_upd_1) -  mean(W_0*(Y-hat_m_0)*weights_upd_0)

    results[b,] <- c(tau_db,tau_pred)
    print(b)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
```

```
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
```

```
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
## [1] 112
## [1] 113
## [1] 114
## [1] 115
## [1] 116
## [1] 117
## [1] 118
## [1] 119
## [1] 120
## [1] 121
## [1] 122
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
## [1] 128
## [1] 129
## [1] 130
## [1] 131
## [1] 132
## [1] 133
## [1] 134
## [1] 135
## [1] 136
## [1] 137
## [1] 138
## [1] 139
```

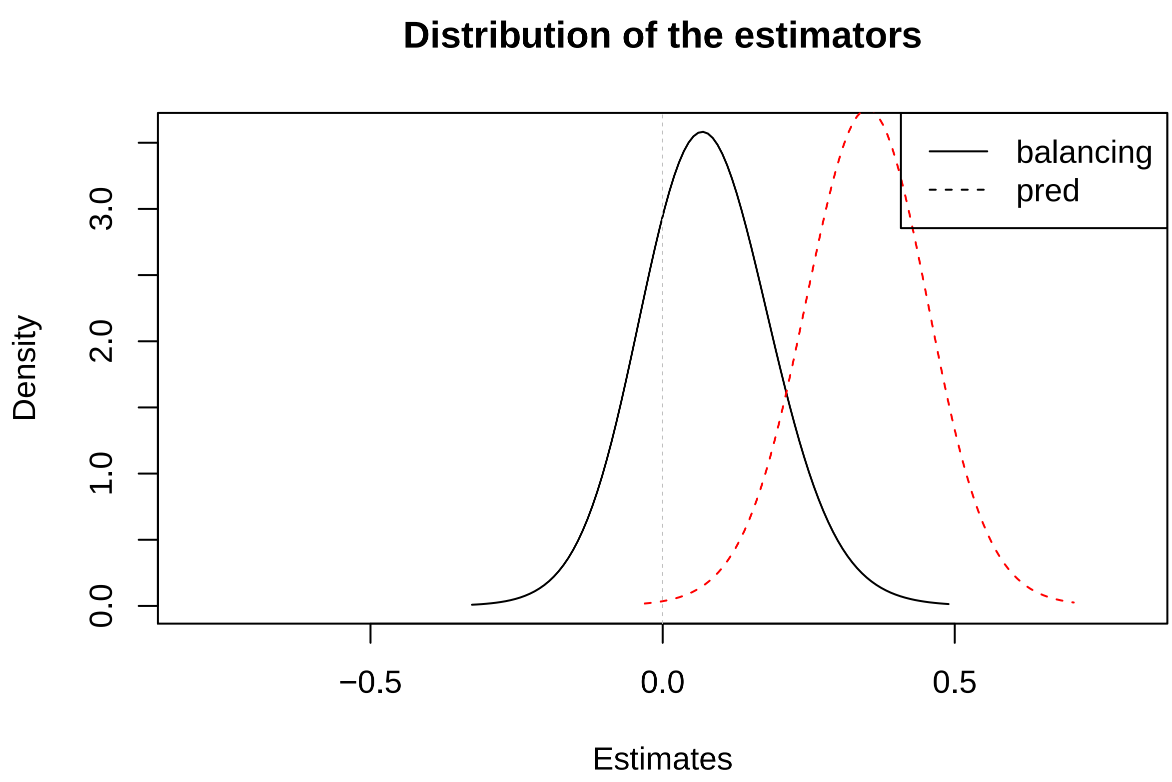
```
## [1] 140
## [1] 141
## [1] 142
## [1] 143
## [1] 144
## [1] 145
## [1] 146
## [1] 147
## [1] 148
## [1] 149
## [1] 150
## [1] 151
## [1] 152
## [1] 153
## [1] 154
## [1] 155
## [1] 156
## [1] 157
## [1] 158
## [1] 159
## [1] 160
## [1] 161
## [1] 162
## [1] 163
## [1] 164
## [1] 165
## [1] 166
## [1] 167
## [1] 168
## [1] 169
## [1] 170
## [1] 171
## [1] 172
## [1] 173
## [1] 174
## [1] 175
## [1] 176
## [1] 177
## [1] 178
## [1] 179
## [1] 180
## [1] 181
## [1] 182
## [1] 183
## [1] 184
## [1] 185
## [1] 186
## [1] 187
## [1] 188
## [1] 189
## [1] 190
## [1] 191
## [1] 192
## [1] 193
```

```
## [1] 194
## [1] 195
## [1] 196
## [1] 197
## [1] 198
## [1] 199
## [1] 200
```

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



plot(bal_est,main = 'Distribution of the estimators', xlim = c(-0.8,0.8),
     col = 'black',lty = 1,type = 'l',xlab = 'Estimates',ylab = 'Density')
lines(pred_est, col = 'red',lty = 2)
abline(v =0, lty = 2,lwd = 0.5,col = 'grey')
legend('topright',col <- c('black','red',), lty = c(1,2),
       legend = c('balancing','pred'))
```




```
rmse_res <- round(sqrt(colMeans((results-tau_het)^2)),2)
names(rmse_res) <- c('bal','pred')
```

The doubly robust estimator (black) has a lower bias than the simple estimator tau_pred (red).