Linear model and Lasso linear model simulation

Eliza Chai

06/09/2022

In this problem, you will use the package `simulator` to perform the following simulation study.

Let X be a nÖp data matrix where each entry is generated from a normal distribution with mean 0 and standard deviation 1. Let y be the vector of responses

$y = X\beta + \epsilon$,

with β = (0,1,2,0,0) (and therefore $p = 5$) and ϵ a n-vector where each entry is generated from a normal distribution with mean 0 and standard deviation σ.

We decide to approach the estimation of β with the following two methods:

- Simple linear model (`lm`)
- Lasso linear model: Suppose now you want to take advantage of the fact that some entries of the unknown vector β are zero, i.e. β is sparse. You therefore decide to apply a lasso linear model for the estimation of β. Given a data matrix `x` and an output vector `y`.

```
devtools::install_github("jacobbien/simulator")
library(simulator)
library(glmnet)
cv.out <- cv.glmnet(x,y,alpha=1,nfolds=5) # Cross validated choice of the penalty
optimal_lambda = cv.out$lambda[which.min(cv.out$cvm)]
beta_est = as.numeric(glmnet(x,y,alpha=1, lambda = optimal_lambda)$beta) # Run model
beta_est #This is the lasso estimate of beta
```

For the two methods above, you want evaluate the estimation accuracy using the two metrics:

- L2 norm: $\Sigma_j(\beta_j - \hat{\beta}_j)^2$
- Support recovery, i.e. the proportion of entries of β that were correctly estimated to be equal to or different from zero (in other words, the proportion of entries $j = 1, ..., p$ for which the following statement holds true: ($\beta_j > 0$ and $\hat{\beta}_j > 0$) or ($\beta_j = 0$ and $\hat{\beta}_j = 0$))

Run a simulation study with $n = 40$ and $\sigma^2 = \{1, 4, 7, 10\}$. Plot the results of the estimation accuracy (L2 norm and support recovery) in function of $\sigma^2 = \{1, 4, 7, 10\}$.

Comment on the results.

```
library(purrr)
library(simulator)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-6
```

```r
make_mv_linear_model <- function(n, beta, sigma_sq)
{
  new_model(          # Model constructor
    name = "mv_lm",
    label = sprintf("n = %s, beta = %s, sigma_sq = %s", n, paste(beta, collapse = " "), sigma_sq),
    params = list(beta = beta, sigma_sq = sigma_sq, n = n),
    simulate = function(n, beta, sigma_sq, nsim)
    {
      sim_list =  map(1:nsim, function(ii){
        p = length(beta)
        x <- matrix(rnorm(n*p, mean = 0, sd = 1), nrow = n, ncol = p)
        y <- x %*% beta + rnorm(n, 0, sqrt(sigma_sq))
        list("x" = x, "y" = y)})

      return(sim_list)
    }
  )
}


lse <- new_method("lse", "LSE  w/o intrcpt",
                  method = function(model, draw) {
                    yy <- draw$y
                    xx <- draw$x
                    fit <- lm(yy ~ xx - 1)
                    list(betahat = fit$coef)
                  })


lasso <- new_method("lasso", "LASSO  w/o intrcpt",
                    method = function(model, draw) {
                      yy <- draw$y
                      xx <- draw$x

                      cv.out <- cv.glmnet(xx,yy,alpha=1,nfolds=5)    # Cross validated choice of the pen
                      optimal_lambda = cv.out$lambda[which.min(cv.out$cvm)]
                      beta_est = as.numeric(glmnet(xx,yy,alpha=1, lambda = optimal_lambda)$beta)   # Run

                      list(betahat = beta_est)
                    })

sq_err <- new_metric("l2",
             "Squared error",
             metric = function(model, out) {
               sum((out$betahat - model$beta)^2)    # beta is a scalar
             })

prop_est <- new_metric("support",
             "support recovery",
             metric = function(model, out) {
```

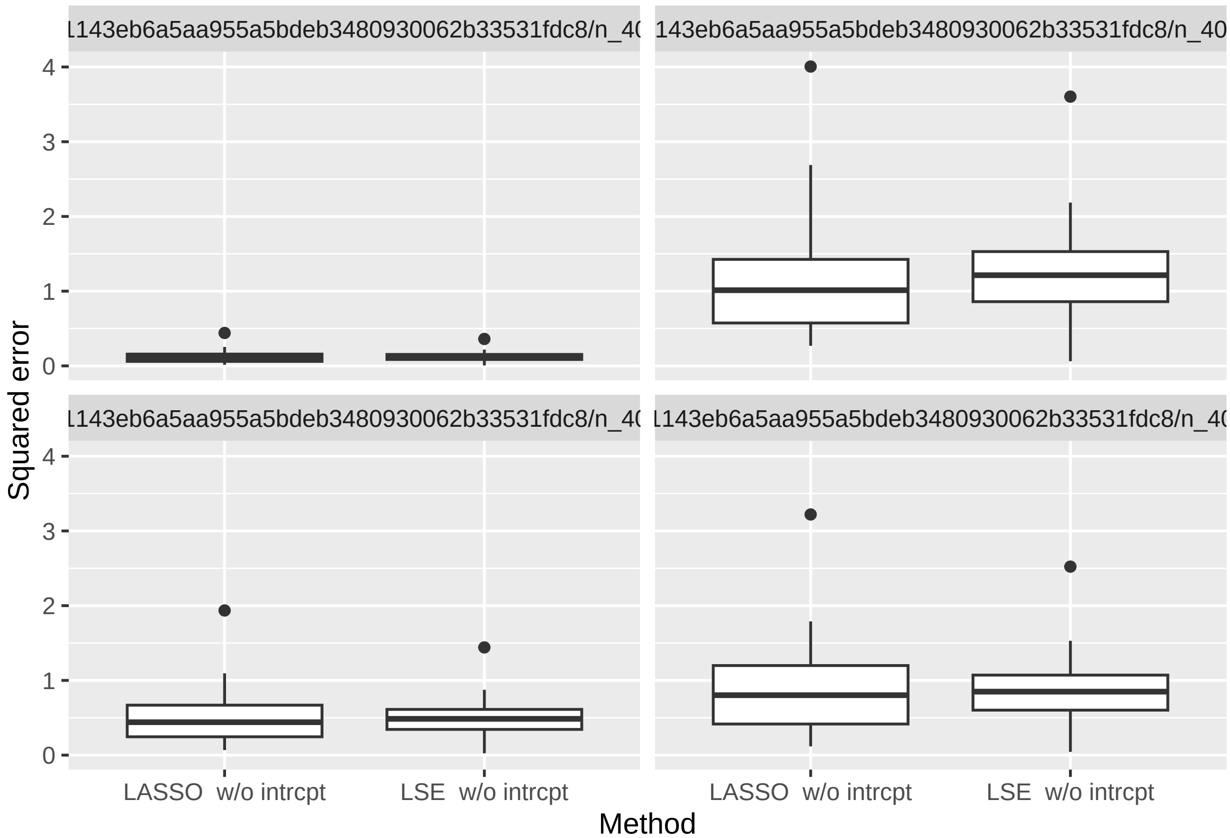
```r
              p = length(model$beta)
              correct= 0
              for (i in 1:p){
                if (model$beta[i] == 0 & out$betahat[i] == 0){
                  correct = correct + 1
                }
                else if (model$beta[i] > 0 & out$betahat[i] > 0){
                  correct = correct + 1
                }
              }
              correct/p
            })


sim <- new_simulation("ls_vs_lasso", "l2 vs support") %>%
  generate_model(make_mv_linear_model,                      # Specify data generation function
                n = 40,                                     # Data generation parameter 1
                sigma_sq = as.list(c(1,4,7,10)),
                beta = c(0,1,2,0,0),
                vary_along = c("sigma_sq")) %>%
  simulate_from_model(nsim = 15)  %>%
  run_method(list(lse,lasso)) %>%
  evaluate(list(sq_err, prop_est))
```

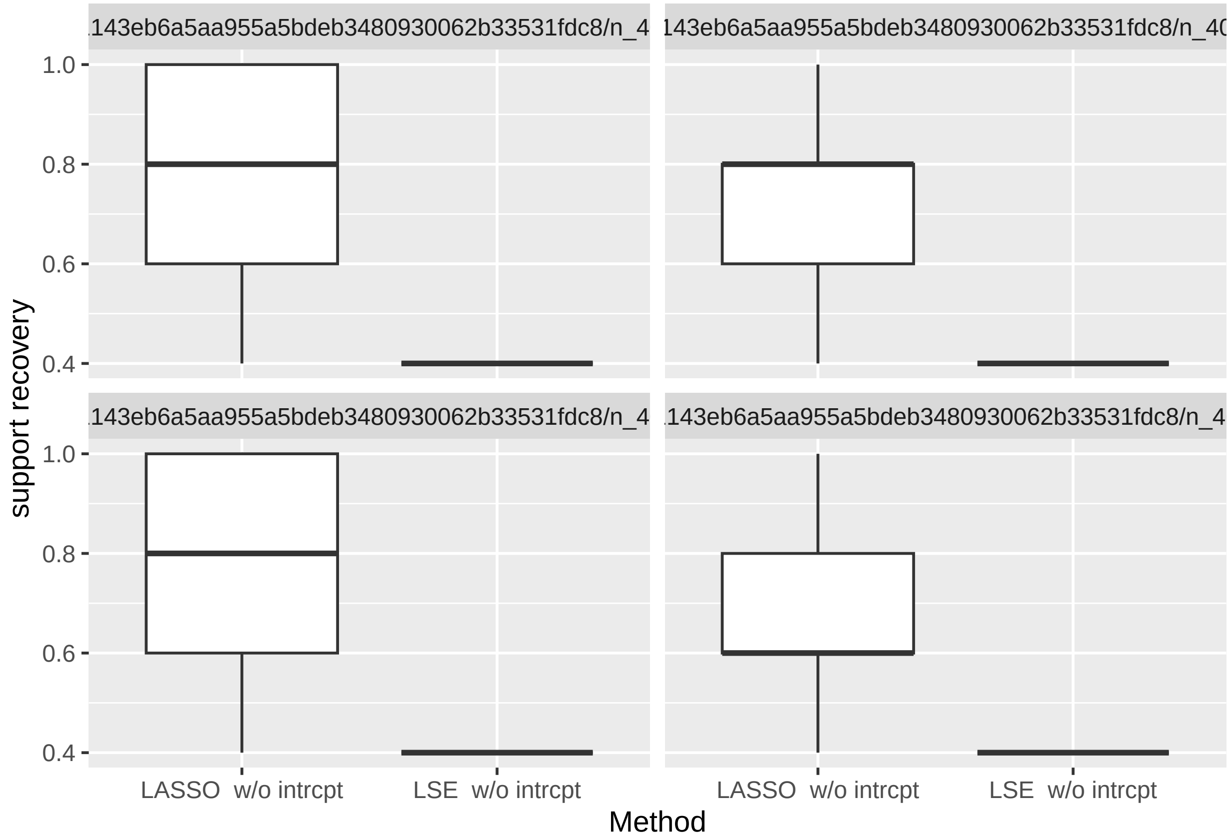
```
## ..Created model and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/sigma_sq_1/mode
## ..Created model and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/sigma_sq_4/mode
## ..Created model and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/sigma_sq_7/mode
## ..Created model and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/sigma_sq_10/mod
## ..Simulated 15 draws in 0 sec and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/s
## ..Simulated 15 draws in 0 sec and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/s
## ..Simulated 15 draws in 0 sec and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/s
## ..Simulated 15 draws in 0 sec and saved in mv_lm/beta_1143eb6a5aa955a5bdeb3480930062b33531fdc8/n_40/s
## ..Performed LSE  w/o intrcpt in 0 seconds (on average over 15 sims)
## ..Performed LASSO  w/o intrcpt in 0.01 seconds (on average over 15 sims)
## ..Performed LSE  w/o intrcpt in 0 seconds (on average over 15 sims)
## ..Performed LASSO  w/o intrcpt in 0.01 seconds (on average over 15 sims)
## ..Performed LSE  w/o intrcpt in 0 seconds (on average over 15 sims)
## ..Performed LASSO  w/o intrcpt in 0.01 seconds (on average over 15 sims)
## ..Performed LSE  w/o intrcpt in 0 seconds (on average over 15 sims)
## ..Performed LASSO  w/o intrcpt in 0.01 seconds (on average over 15 sims)
## ..Evaluated LSE  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LASSO  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LSE  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LASSO  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LSE  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LASSO  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LSE  w/o intrcpt in terms of
```

```
## Squared error, support recovery, Computing time (sec)
## ..Evaluated LASSO  w/o intrcpt in terms of
## Squared error, support recovery, Computing time (sec)
```

```r
sim %>% plot_eval(metric_name = "l2")
```




```r
sim %>% plot_eval(metric_name = "support")
```



The lasso linear model without intercept has a smaller squared error compared to the linear model without intercept using the L2 norm matrices, meaning better estimation accuracy.

The lasso linear model without intercept has a higher support recovery compared to the linear model without intercept, meaning better estimation accuracy.