ST509_HW3_2_2024020409

Hwijun Kwon

2024-04-11

1

(a)

Load Dataset

```
train <- matrix(scan("train.txt"), 500, 51)
test  <- matrix(scan("test.txt"), 500, 51)
x <- train[,-51] ; y <- train[,51]
x.test <- test[,-51] ;  y.test <- test[,51]
```

By my code

```
S <- function(z, lambda){
    sign(z) * max(abs(z) - lambda, 0)
}

cd.elastic <- function(x, y, alpha, lambda){
  # CD algorithm for lasso    # marginal standardization of x
  z <- scale(x)
  m <- attr(z, "scaled:center")  # save original mean of X
  s <- attr(z, "scaled:scale")   # save original scale of X
  u <- (y - mean(y))
  p <- ncol(z)
  n <- length(y)
  # initialization
  beta <- coef(lm(y ~ z - 1)); r <- y - z %*% beta

  for (iter in 1:100) {
    new.beta <- beta
    for (j in 1:p) {
      temp <- beta[j] + crossprod(z[,j], r)/n
      new.beta[j] <- S(temp, (lambda * alpha) / s[j])/ (1+ lambda * (1 - alpha)) # S(temp, (alpha +))
      r <- r - (new.beta[j] - beta[j]) * z[,j]
    }
    delta <- max(abs(new.beta - beta))
    if (delta < 1.0e-3) break
    beta <- new.beta
  }
  beta <- new.beta/s
  c(beta)
}
```

Grid Search

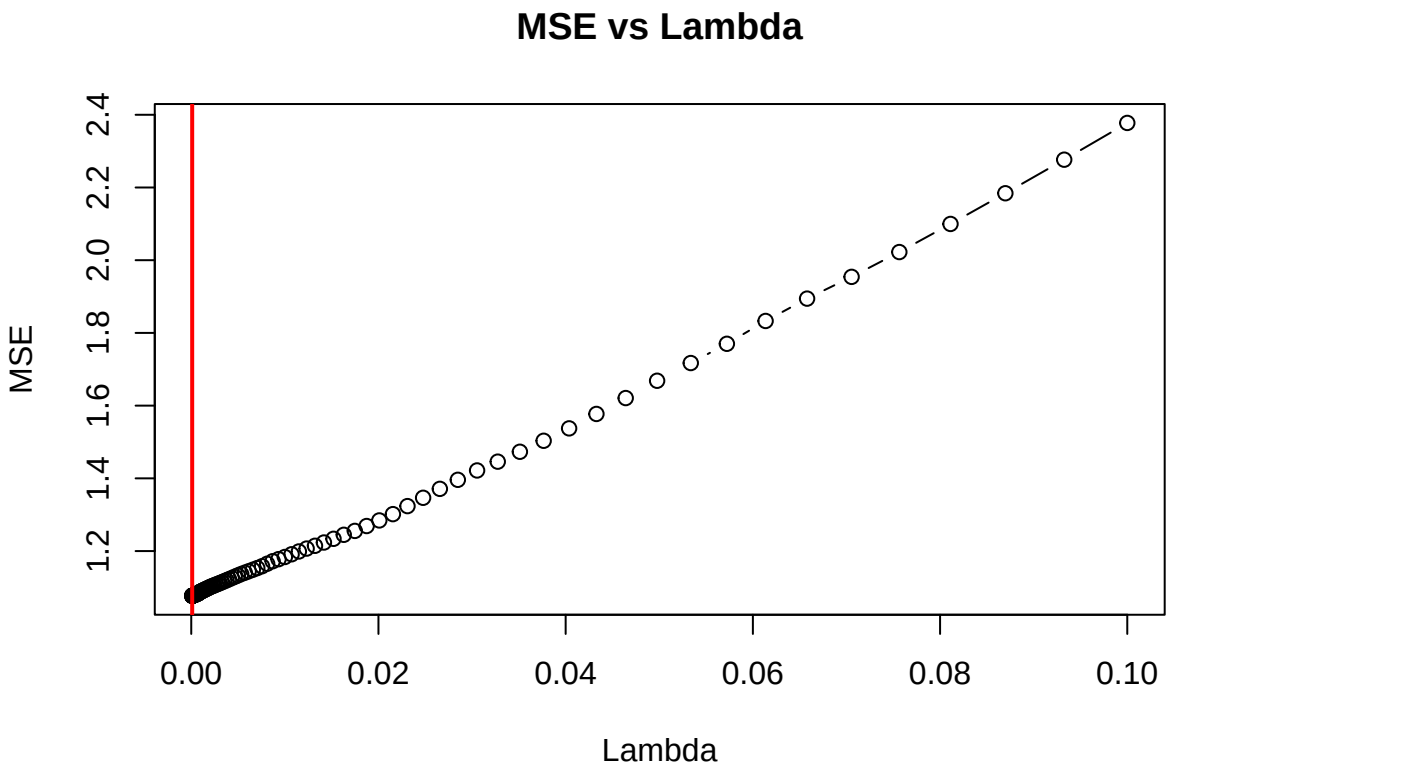
```
lambda_values <-10^seq(-4, -1, length.out =100)
mse_results <- numeric(length(lambda_values))
num_beta <- numeric(length(lambda_values))
idx = 1
for (i in seq_along(lambda_values)) {
  beta_hat = cd.elastic(x, y, 0.5, lambda = lambda_values[i])
  y_pred = x.test %*% beta_hat # (500,50) %*% (50.1)
  mse = mean((y.test-y_pred)**2)
  mse_results[i] = mse
  num_beta[i] = sum(beta_hat != 0)

}

# Lambda/MSE PLOT
results = data.frame(Lambda = lambda_values , MSE = mse_results, num_beta = num_beta)
min_mse_index <- which.min(results$MSE)
min_lambda <- results$Lambda[min_mse_index]
cat("Min MSE", min(results$MSE))
```

```
## Min MSE 1.076858
```

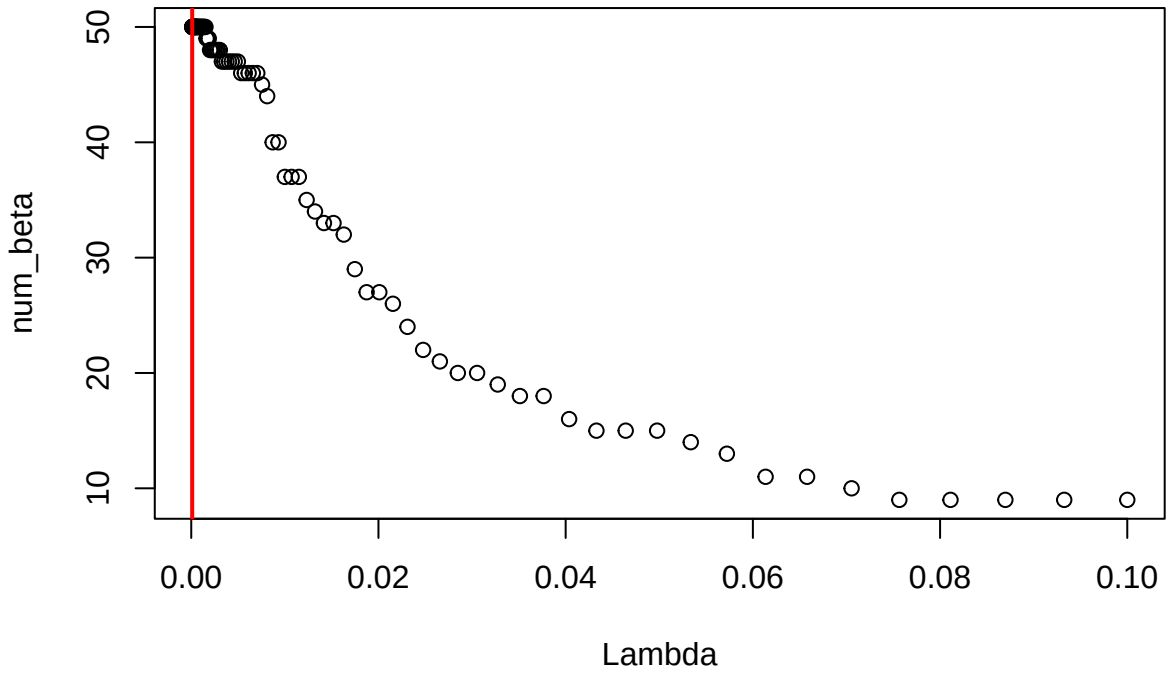
```
cat("\nBest Lambda", min_lambda )
```

```
##
## Best Lambda 1e-04
```

```
# Plot MSE VS Lambda
plot(results$Lambda, results$MSE, type = 'b', xlab = "Lambda", ylab = "MSE", main = "MSE vs Lambda")
abline(v = min_lambda, col = 'red', lwd = 2)
```



```r
# Plot Lamda vs Num Coeff
plot(data.frame(Lambda = lambda_values , num_beta = num_beta))
abline(v = min_lambda, col = 'red', lwd = 2)
```



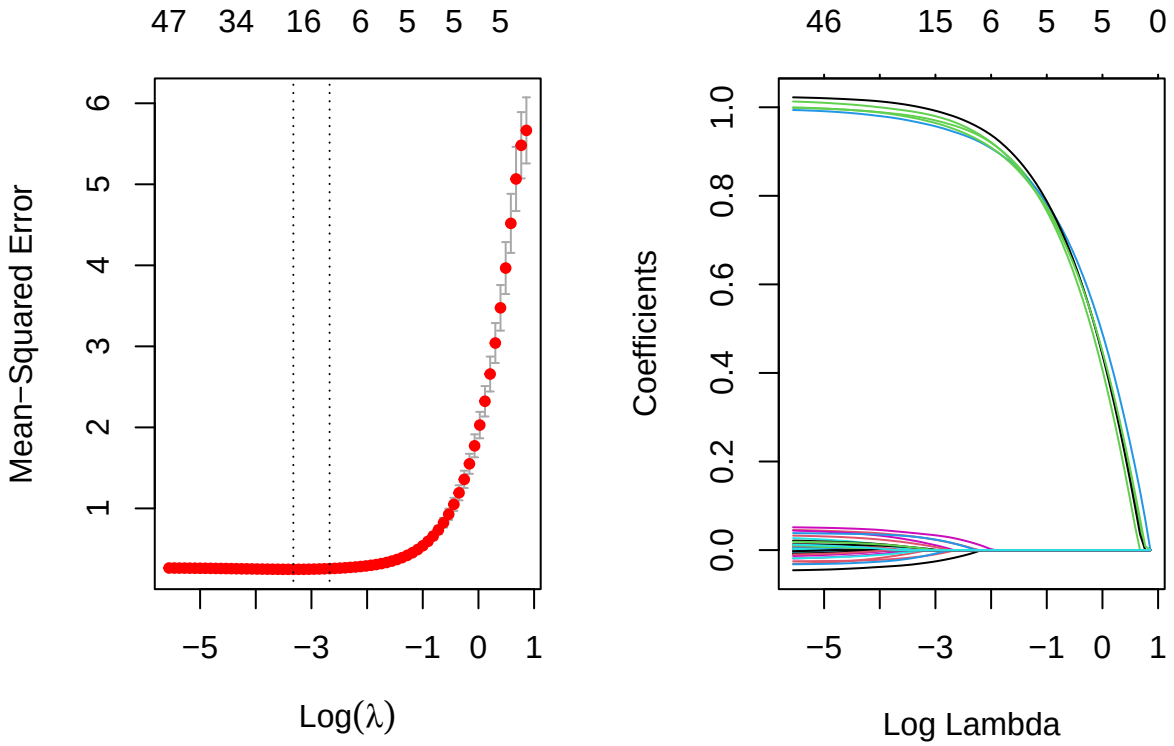
By glm function

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

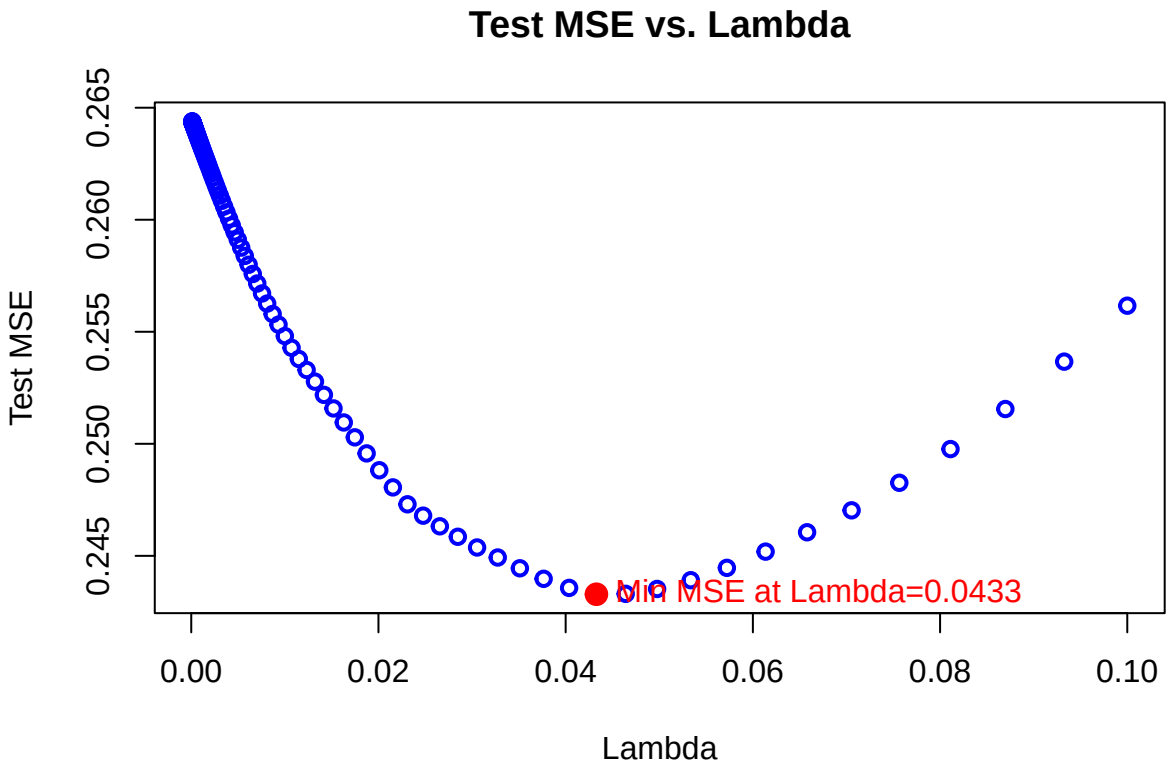
```r
cv_model <- cv.glmnet(x, y, alpha=0.5, type.measure="mse")

par(mfrow = c(1, 2))

plot(cv_model)
plot(cv_model$glmnet.fit, "lambda")
```




```
lambda <- 10^seq(-4, 0, length.out = 100)
mse_list <- numeric(length(lambda_values))

for (i in seq_along(lambda_values)) {
  model <- glmnet(x, y, alpha=0.5, lambda=lambda_values[i])
  y_pred<- predict(model, s=lambda[i], newx=x.test)
  mse_list[i] <- mean((y.test - y_pred)^2)
}

min_mse_idx <- which.min(mse_list)
lambda_min <- lambda_values[min_mse_idx]
min_mse <- mse_list[min_mse_idx]

cat("\nLambda_Min:", lambda_min, "\nMin_MSE:", min_mse)
```

```
##
## Lambda_Min: 0.04328761
## Min_MSE: 0.2432862
```

```
# MSE vs Lambda graph
plot(lambda_values, mse_list, col = "blue", lwd = 2,
     xlab = "Lambda", ylab = "Test MSE", main = "Test MSE vs. Lambda")
points(lambda_min, min_mse, col = "red", pch = 19, cex = 1.5)
text(lambda_min, min_mse, labels = sprintf("Min MSE at Lambda=%.4f", lambda_min), pos = 4, col = "red")
```



###

(b) Report the non-zero regression coefficients setimates you obtained

By My Code

BY glm

```
model <- glmnet(x, y, alpha = 0.5 , lambda = lambda_min)
coefficients_lambda_min <- coef(model, s = "lambda.min")
coefficients_lambda_min <- as.matrix(coefficients_lambda_min)
nonzero_coefficients_lambda_min <- coefficients_lambda_min[coefficients_lambda_min[, 1] != 0, , drop = F
print(paste("Non-zero Coefficients at lambda.min:", lambda_min, "when alpah is 0.5"))
```

```
## [1] "Non-zero Coefficients at lambda.min: 0.0432876128108306 when alpah is 0.5"
```

```
print(nonzero_coefficients_lambda_min)
```

```
##                       s1
## (Intercept)  1.052096894
## V8          -0.028185102
## V10          0.974205877
## V17          0.961187616
## V20          0.996393938
## V21         -0.012458379
## V22          0.983955422
## V27          0.002910686
## V28          0.026238911
## V33          0.014827350
## V34          0.000579792
## V39          0.035293607
## V41          0.010018713
## V42          0.968798164
```

```
## V43         -0.011783858
## V48          0.002040429
## V49          0.026391564
```