HW6_lars_implementation

Yutaro Yamada

December 16, 2015

Loading diabetes data

```
# load data
library(lars)
```

```
## Loaded lars 1.2
```

```
data(diabetes)
dd =diabetes
X = dd$x
X = scale(X)
y = dd$y
y = y - mean(y)
```

lars function

```
myLars <- function(X,y){
  n = nrow(X)
  p = ncol(X)
  # for storing beta values
  beta_mat <- matrix(0, nrow=p, ncol=12)
  # init
  b_hat = rep(0, min(n,p))
  A <- list()
  sequence <- list()
  lam = .Machine$integer.max
  user_defined_lam = 20
  count = 1
  first = TRUE

  while(1){
    cat(count)
    # update C_j(lam_k)
    c = rep(0, min(n,p))
    for(j in 1:p){
      c[j] = t(X[,j]) %*% (y - X %*% b_hat)
    }
    # Update Active Set
    if(first){
      first = FALSE
      max_id = which(max(abs(c)) == abs(c)) # taking the id that maximizes X_j'y
```

```
  lam_k = max(abs(c))
  sequence[length(sequence) + 1] <- max_id
  A[length(A) + 1] <- max_id # Now Active set is {1} (or {max_id})
  beta_mat[,count] <- b_hat
}else{
  if(max_id > 0){
    sequence[length(sequence) + 1] <- max_id
    A[length(A) + 1] <- max_id
    beta_mat[,count] <- b_hat
  }else{
    sequence[length(sequence) + 1] <- -A[[keep_id_lam_tilda]]
    A[[keep_id_lam_tilda]] <- NULL
    beta_mat[,count] <- b_hat
  }
}

# Stopping criteria  # will implement later
if(count == 12){break}

# Construct Z matrix
Z = matrix(0, nrow=n, ncol=length(A))
for(i in 1 : length(A)){
  Z[,i] = sign(c[A[[i]]])*X[,A[[i]]]
}
v = solve(t(Z) %*% Z) %*% rep(1, length(A))


# Update lambda
if(length(A)!=1){ # if not first time.
  lam_k = lam
}
lam_hat = 0
for(j in 1 : p){
  if( (j %in% A) == FALSE){
    alpha = lam_k*t(X[,j])%*%Z%*%v
    gamma = t(X[,j])%*%Z%*%v
    #if(c[j] > 0){
    temp1 = (c[j] - alpha) / (1 - gamma)
    #}else{
    temp2 = (-c[j] + alpha) / (1 + gamma)
    if(temp1 > lam_k){ temp1 = 0}
    if(temp2 > lam_k){ temp2 = 0}

    if(temp1 > temp2){
      temp= temp1; flag1 = TRUE; flag2 = FALSE
    }else{
      temp = temp2; flag2 = TRUE; flag1 = TRUE
    }
    #temp = max(temp1, temp2)
    if(temp > 0 & temp > lam_hat & temp < lam_k){
        lam_hat = temp
        keep_id_lam_hat = j
```

```
            #if((flag1 & (lam_hat > lam_k - c[j]/gamma ))){
            #  cat("ok flag1"); cat(j)
            #}
            #if((flag2 & (lam_hat <= lam_k - c[j]/gamma ))){
            #  cat("ok flag2"); cat(j)
            #}
        }
      }
    }
    #cat(lam_hat)


    lam_tilda = 0
    for(jj in 1 : length(A)){
      temp = lam_k + b_hat[A[[jj]]]/(v[jj]*sign(c[A[[jj]]]))
      if(temp < lam_k & temp > lam_tilda){
        lam_tilda = temp
        keep_id_lam_tilda = jj # A[[j]]
      }
    }
    #cat(lam_tilda)


    # Update lamda
    if(lam_hat > lam_tilda){
      lam = lam_hat
    }else{
      lam = lam_tilda
      #print("FLAG!!!")
    }
    #lam = lam_hat
    #lam_tilda = 0


    # Update b_hat
    for(jj in 1:length(A)){
      b_hat[A[[jj]]] = b_hat[A[[jj]]] + (lam_k - lam)*v[jj]*sign(c[A[[jj]]])
    }

    # Fix indexing
    if(lam_hat > lam_tilda){ # max_id is the candidate id for a new member in Active set
      max_id = keep_id_lam_hat
    }else{
      #delete this id (keep_id_lam_tilda) from Active set
      max_id = -1
    }

    count = count + 1
  }

  return(list(A, as.numeric(sequence), beta_mat))
}
```

Results

```
result <- myLars(X,y)
```

```
## 123456789101112
```

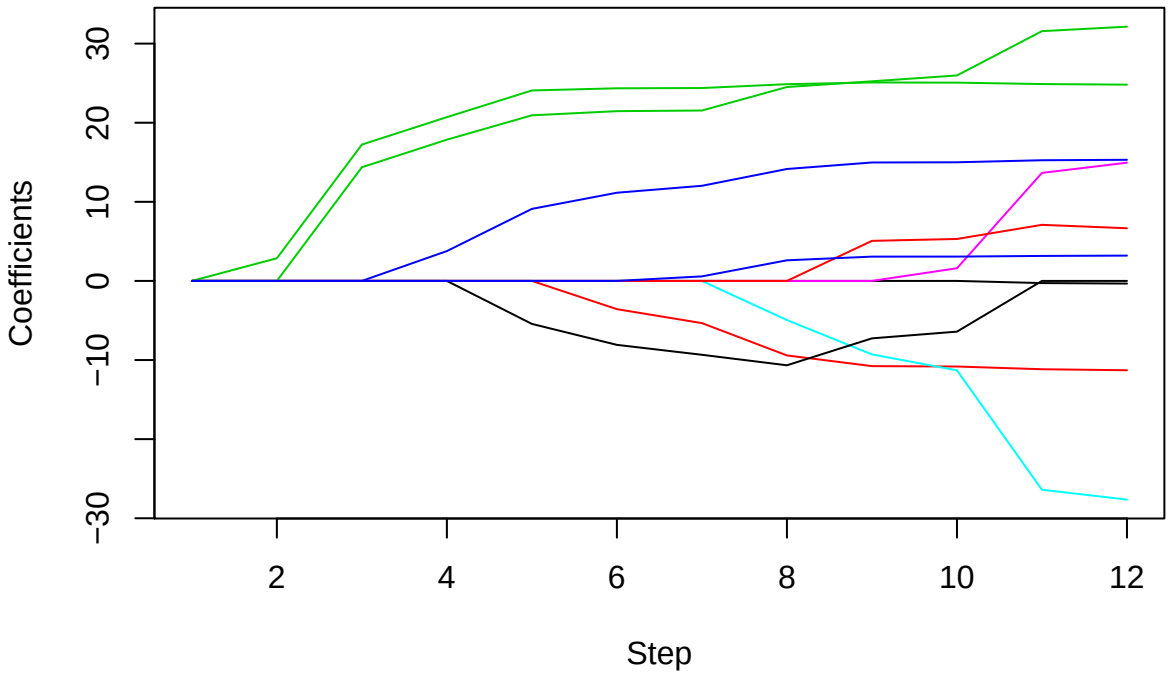
```
# Active set
result[[1]]
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 9
##
## [[3]]
## [1] 4
##
## [[4]]
## [1] 2
##
## [[5]]
## [1] 10
##
## [[6]]
## [1] 5
##
## [[7]]
## [1] 8
##
## [[8]]
## [1] 6
##
## [[9]]
## [1] 1
##
## [[10]]
## [1] 7
```

```
# Variables added/deleted at each step
result[[2]]
```

```
##  [1]  3  9  4  7  2 10  5  8  6  1 -7  7
```

```
# The LASSO plot : step v. coefficients
matplot(seq(1:12), t(result[[3]]), type = "l", lty = 1, xlab="Step", ylab="Coefficients")
```


Coefficients
30
20
10
0
−10
−30
2
4
6
8
10
12
Step