A-6.R

pradyuth

Tue Apr 17 05:55:49 2018

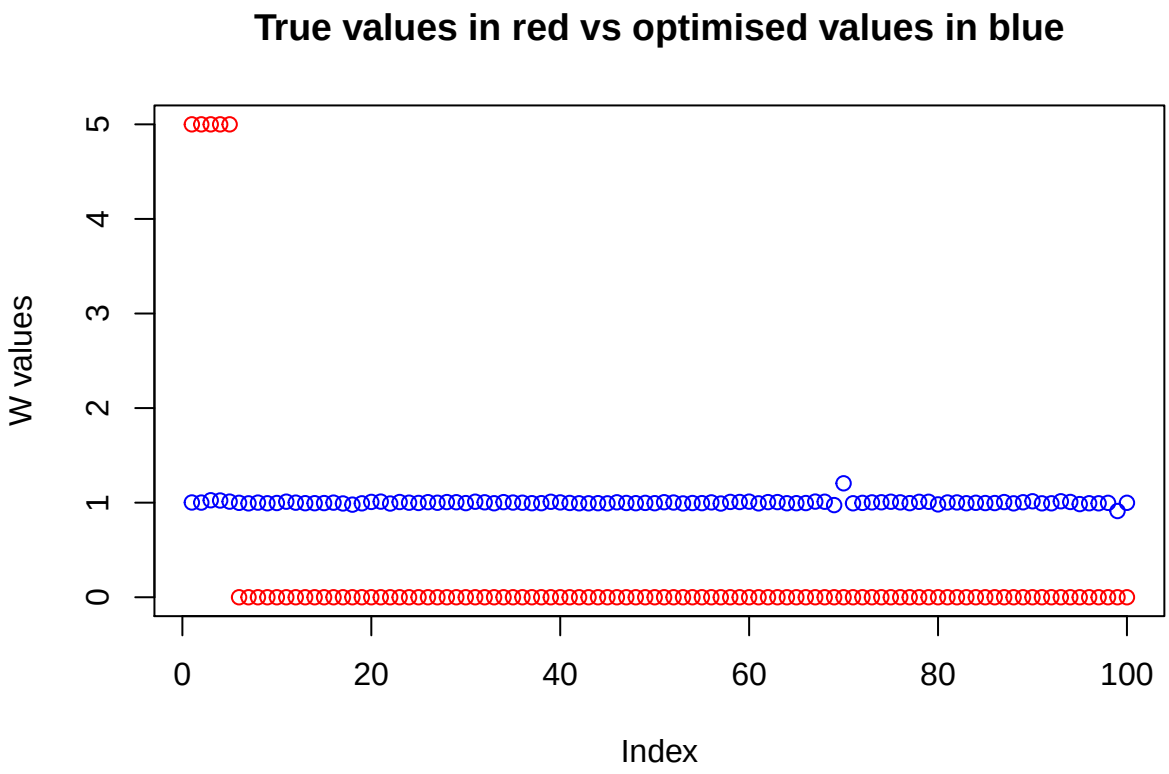
```
#Name : Pradyuth Vangur

#Problem 1: Function for generating training dataset
gen_data <- function(n, p, sparsity, level){
  X <- matrix(rnorm(n*p, mean = 0, sd = 1), nrow = n, ncol = p)
  w <- matrix(rep(0, p), nrow = p, ncol = 1)
  w[1:sparsity,] <- level
  Y <- X%*%w
  return(list(X,Y))
}

#Problem 2: Creating loss function
lasso_loss <- function(w, lambda){
  func_to_optim <- (sum(((y_glbl - (x_glbl %*% w))^2)) + lambda*(sum(abs(w))))
  return(func_to_optim)
}

#Problem 3: Generating training dataset
A <- gen_data(50, 100, 5, 5)
x_glbl <- A[[1]]
y_glbl <- A[[2]]
w_glbl <- matrix(rep(0, 100), nrow = 100, ncol = 1)
w_glbl[1:5,] <- 5

#Problem 4: Using the optim function to find best values of w
w_values <- optim(par = matrix(rep(1,100)),fn = lasso_loss, lambda = 1)
w_values
```

```
## $par
##              [,1]
##   [1,] 1.0018466
##   [2,] 1.0009720
##   [3,] 1.0254984
##   [4,] 1.0229653
##   [5,] 1.0109695
##   [6,] 0.9981940
##   [7,] 0.9926348
##   [8,] 1.0006785
##   [9,] 0.9934843
##  [10,] 0.9963937
##  [11,] 1.0104975
##  [12,] 1.0000799
##  [13,] 0.9942780
##  [14,] 0.9943919
##  [15,] 0.9954104
##  [16,] 1.0002606
##  [17,] 0.9915247
```

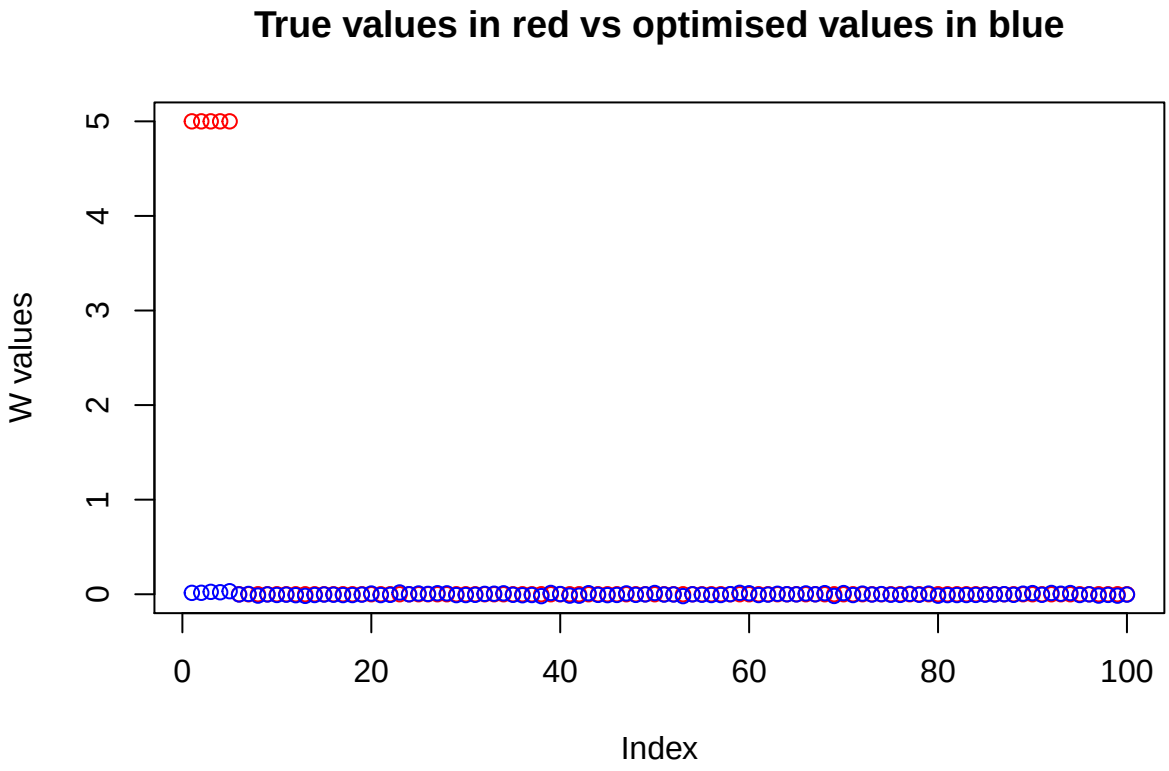
```
##  [18,] 0.9790964
##  [19,] 0.9929949
##  [20,] 1.0078926
##  [21,] 1.0109253
##  [22,] 0.9909123
##  [23,] 1.0067455
##  [24,] 0.9998439
##  [25,] 0.9969811
##  [26,] 1.0051768
##  [27,] 0.9990151
##  [28,] 1.0059689
##  [29,] 1.0044638
##  [30,] 0.9950890
##  [31,] 1.0102392
##  [32,] 1.0036223
##  [33,] 0.9937854
##  [34,] 1.0054748
##  [35,] 1.0010975
##  [36,] 0.9994287
##  [37,] 0.9943006
##  [38,] 0.9947742
##  [39,] 1.0093559
##  [40,] 1.0025798
##  [41,] 0.9970097
##  [42,] 0.9934312
##  [43,] 0.9931918
##  [44,] 0.9946505
##  [45,] 0.9928041
##  [46,] 1.0035446
##  [47,] 0.9965829
##  [48,] 0.9944726
##  [49,] 0.9973826
##  [50,] 0.9942567
##  [51,] 1.0045128
##  [52,] 1.0015173
##  [53,] 0.9919020
##  [54,] 0.9961881
##  [55,] 0.9950281
##  [56,] 1.0027040
##  [57,] 0.9909474
##  [58,] 1.0077981
##  [59,] 1.0069712
##  [60,] 1.0106338
##  [61,] 0.9925698
##  [62,] 1.0055054
##  [63,] 1.0059126
##  [64,] 0.9939095
##  [65,] 0.9942504
##  [66,] 0.9958696
##  [67,] 1.0113500
##  [68,] 1.0098369
##  [69,] 0.9742572
##  [70,] 1.2046098
##  [71,] 0.9935402
```

```
##  [72,] 0.9977869
##  [73,] 1.0027798
##  [74,] 1.0040432
##  [75,] 1.0097834
##  [76,] 1.0032011
##  [77,] 0.9948754
##  [78,] 1.0086315
##  [79,] 1.0094326
##  [80,] 0.9816534
##  [81,] 1.0019377
##  [82,] 1.0023586
##  [83,] 0.9937349
##  [84,] 0.9984915
##  [85,] 0.9957972
##  [86,] 0.9957704
##  [87,] 1.0061391
##  [88,] 0.9927452
##  [89,] 1.0043688
##  [90,] 1.0147911
##  [91,] 0.9929085
##  [92,] 0.9931986
##  [93,] 1.0147492
##  [94,] 1.0073853
##  [95,] 0.9847341
##  [96,] 0.9937626
##  [97,] 0.9933387
##  [98,] 0.9979548
##  [99,] 0.9115022
## [100,] 0.9986035
##
## $value
## [1] 6076.447
##
## $counts
## function gradient
##      501       NA
##
## $convergence
## [1] 1
##
## $message
## NULL
```

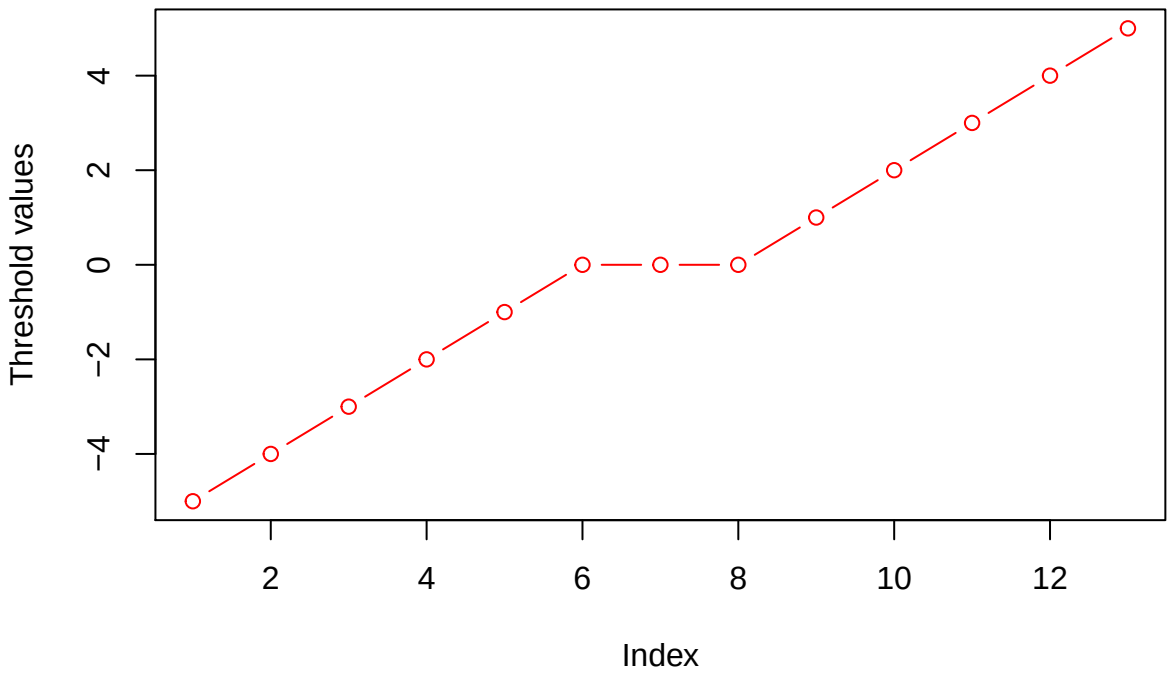
```
plot(w_glbl, type = "p", col = "red", ylab = "W values",
     main = "True values in red vs optimised values in blue")
points(w_values$par, col = "blue")
```




```r
#Problem 5: Using the optim function to find the best values of w and lambda
input <- function(ip_1){
  a <- lasso_loss(ip_1[-1], ip_1[1])
  return(a)
}

w_values_1 <- optim(par = c(1, matrix(rep(0,100))),fn = input)

plot(w_glbl, type = "p", col = "red", ylab = "W values",
     main = "True values in red vs optimised values in blue")
points(w_values_1$par[-1], col = "blue")
```




```
#Coordinate descent
#Problem 1: Writing soft threshold function
soft_threshold <- function(w, th){
  value <- ifelse((abs(w) < th), 0, sign(w)*(abs(w) - th))
  return(value)
}

plot(soft_threshold(-6:6, 1), type = "b", col = "red", ylab = "Threshold values")
```




```r
#Problem 2: Lasso1d function
lasso1d <- function(x, y, lambda){
  w <- ((t(y) %*% x)/ (t(x) %*% x))
  th <- (lambda/ (t(x)%*%x))
  return(soft_threshold(w, th))
}



#Problem 3: Capturing the residuals
get_residual <- function(w, dim, X, Y){
  w[dim] <- 0
  Y_pred <- X %*% w
  Y_res <- (Y - Y_pred)
  return(Y_res)
}


#Problem 4:
#value stands for the estimated percentage change
cor_desc <- function(w, lambda, value, X, Y){
  w_new <- w*(value + 1)
  w_old <- w
  while(abs(((w_new - w_old)*100)) >= abs(w_old)*value){
    w_old <- w_new
    for(i in 1:length(w)){
```

```
    x_ele <- X[,i]
    residual <- get_residual(w_new, i, X, Y)
    w_calc <- ((t(x_ele) %*% residual) / (t(x_ele) %*% x_ele))
    th <- (lambda/ (t(x_ele)%*%x_ele))
    w_new[i] <- soft_threshold(w_calc, th)
   }
  }
  return(w_new)
}


#Problem 5:
cor_desc(rep(0,100), 1, 2, x_glbl, y_glbl)
```

```
## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

##   [1]  3.068210644  3.879931293  4.769325144  4.608672005  4.827649647
##   [6]  0.024974164  0.330309526 -0.378946800 -0.416163975  0.159013201
##  [11]  0.316568774  0.188897723  0.003441634 -0.043692361 -0.107263654
##  [16] -0.208826894  0.000000000 -0.119147844 -0.132473189 -0.064809812
##  [21] -0.043015810 -0.024017400  0.000000000 -0.282013935  0.217689307
##  [26]  0.243285424  0.081519725 -0.049292766  0.003975943  0.000000000
##  [31]  0.213610738 -0.032169867  0.105479471  0.500435475 -0.062158328
##  [36]  0.000000000 -0.181155256 -0.121642699  0.182064682  0.000000000
##  [41]  0.000000000 -0.220441289  0.080710652  0.000000000 -0.285165120
##  [46] -0.119090714  0.000000000 -0.190392652  0.140667845  0.323371679
##  [51]  0.000000000 -0.129590055  0.000000000 -0.344172108  0.031753454
##  [56] -0.530286516 -0.192676347 -0.134918168  0.332550179 -0.185403881
##  [61] -0.317311126  0.000000000  0.058664875  0.000000000 -0.185665311
##  [66]  0.373249697 -0.163650456  0.069498411 -0.289898029 -0.230709559
##  [71]  0.000000000  0.106745939  0.003490672  0.286418865  0.314486181
##  [76]  0.020862440 -0.029401434  0.000000000 -0.250236356 -0.201728332
```

```
##  [81]  0.463270350 -0.151287995 -0.105154239  0.267163078  0.000000000
##  [86]  0.062914520  0.039518529 -0.070265831  0.040415241  0.066166205
##  [91]  0.005689732  0.214518039  0.029533335  0.013765694 -0.012850955
##  [96] -0.132911813  0.000000000 -0.341974029 -0.201953629 -0.140954869
```

```r
#We observe that the values obtained from the cor_desc is
#way better than optim function values

#Problem 6:
L2 <- list(rep(0,10))
j <- 1
for(i in seq(0,45,5)){
  L2[[j]] <- cor_desc(rep(0,100), 1, 50, x_glbl[(i+1):(i+5), ], y_glbl[(i+1):(i+5), ])
  j <- j + 1
}
```

```
## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used
```

```
## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used

## Warning in while (abs(((w_new - w_old) * 100)) >= abs(w_old) * value) {:
## the condition has length > 1 and only the first element will be used
```

```r
L2_error <- rep(0,10)
for(i in 1:10){
  L2_error[i] <- sqrt(abs(sum(L2[[i]]^2 - w_glbl^2)))
```

```
}

plot(seq(0,45,5), L2_error, xlab = seq(0,45,5), ylab = "L2_error", type = "b",
     main = "L2 error", col = "blue", xlim = c(1, 50))
```