Logistic-Lasso Coordinate Descent Algorithm

Jimmy Kelliher (UNI: jmk2303)

2022-03-20

Theory

Lemma 1. Consider the optimization problem

$$\min_{x \in \mathbb{R}} \left\{ \frac{1}{2}(x-b)^2 + c|x| \right\}$$

for $b \in \mathbb{R}$ and $c \in \mathbb{R}_{++}$. It follows that the minimizer is given by

$$\hat{x} = S(b, c),$$

where S is the soft-thresholding operator.

Lemma 2. Consider the optimization problem

$$\min_{\beta_k \in \mathbb{R}} \left\{ \frac{1}{2n} \sum_{i=1}^{n} w_i \left(z_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\}$$

for some $k \in \{1, \ldots, p\}$. It follows that the minimizer is given by

$$\hat{\beta}_k = \left(\sum_{i=1}^{n} w_i x_{ik}^2 \right)^{-1} \sum_{i=1}^{n} w_i x_{ik} \left(z_i - \sum_{j \neq k} \beta_j x_{ij} \right).$$

Lemma 3. With $\hat{\beta}_k$ defined as above,

$$\min_{\beta_k \in \mathbb{R}} \left\{ \frac{1}{2n} \sum_{i=1}^{n} w_i \left(z_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} = \min_{\beta_k \in \mathbb{R}} \left\{ \frac{1}{2}(\beta_k - \hat{\beta}_k)^2 + \left(\frac{1}{n} \sum_{i=1}^{n} w_i x_{ik}^2 \right)^{-1} \lambda |\beta_k| \right\}.$$

Proposition. By Lemma 1 and Lemma 3,

$$\arg\min_{\beta_k \in \mathbb{R}} \left\{ \frac{1}{2n} \sum_{i=1}^{n} w_i \left(z_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} = S \left(\hat{\beta}_k, \left(\frac{1}{n} \sum_{i=1}^{n} w_i x_{ik}^2 \right)^{-1} \lambda \right)$$

Praxis

```
data <-
  read_csv("data/breast-cancer.csv", show_col_types = FALSE) %>%
  mutate(diagnosis = 1 * (diagnosis == "M"))
```

Helper Functions

```
# logistic function
logistic <- function(x) 1 / (1 + exp(-x))

# shrinkage function
S <- function(beta, gamma) {
  if(abs(beta) <= gamma) {
    0
  } else if(beta > 0) {
    beta - gamma
  } else {
    beta + gamma
  }
}

# probability adjustment function
p_adj <- function(p, epsilon) {
  if (p < epsilon) {
    0
  } else if(p > 1 - epsilon) {
    1
  } else {
    p
  }
}

# weight adjustment function
w_adj <- function(p, epsilon) {
  if ((p < epsilon) | (p > 1 - epsilon)) {
    epsilon
  } else {
    p * (1 - p)
  }
}
```

Toy Example

```
set.seed(1)
lambda <- 0 #0.0125
epsilon <- 10^(-5)

q    <- 30 - 1
n    <- 1000
```

```
X    <- matrix(rnorm(q * n), c(n, q))
X    <- as.matrix(cbind(rep(1, n), X))
y    <- 1 * (runif(n) > 0.5)

# initialize parameters
beta <- rep(0.25, ncol(X))

outer_term <- 0
outer <- 1
while(outer_term < 1) {

p <- map_dbl(logistic(X %*% beta), p_adj, epsilon)
w <- map_dbl(p, w_adj, epsilon)
z <- X %*% beta + (y - p) / w

  terminate <- 0
  iter <- 1
  while(terminate < 1) {

    beta_old <- beta

    for(k in 1:ncol(X)) {
      x_k    <- X[ , k]
      x_notk <- X[ , -k]
      b_notk <- beta[-k]

      # un-penalized coefficient update
      b_k_temp <- sum(w * (z - x_notk %*% b_notk) * x_k) / sum(w * x_k^2)
      # shrinkage update
      b_k      <- S(b_k_temp, lambda * (k > 1) / mean(w * x_k^2))
      # update beta vector along with other parameters
      beta[k] <- b_k
      #p <- map_dbl(logistic(X %*% beta), p_adj, epsilon)
      #w <- map_dbl(p, w_adj, epsilon)
      #z <- X %*% beta + (y - p) / w
    }

    iter <- iter + 1

    if(iter == 100 | max(abs(beta - beta_old)) < 10^(-12)) {
      terminate <- 1
    }

  }

  print(iter)
  outer <- outer + 1

  if(outer == 100 | iter == 2) {
      print(iter)
      outer_term <- 1
  }
```

```
}
```

```
## [1] 21
## [1] 16
## [1] 14
## [1] 13
## [1] 11
## [1] 6
## [1] 2
## [1] 2
```

```
# true estimates from glmnet
fit <- glmnet(X, y, family = "binomial", standardize = FALSE, lambda = lambda, thresh = 10^-12)

# results
results <- tibble(
    Variable   = 1:length(beta)
  , Jimmy      = beta
  , GLM        = as.vector(glm(y ~ X[ , -1], family = binomial)$coefficients)
  , GLMNET     = as.vector(fit$beta[ , ncol(fit$beta)])
  , Difference = abs(Jimmy - GLMNET)
  , Change     = (Jimmy - GLMNET) / GLMNET
) %>%
  mutate(GLM = na_if(GLM, (lambda != 0) * GLM)) %>%
  filter(Jimmy != 0 | GLMNET != 0)
results %>% knitr::kable()
```

Variable	Jimmy	GLM	GLMNET	Difference	Change
1	-0.0066729	-0.0066729	0.0000000	0.0066729	-Inf
2	0.0074020	0.0074020	0.0074020	0.0000000	0e+00
3	-0.0010686	-0.0010686	-0.0010686	0.0000000	-2e-07
4	0.0040323	0.0040323	0.0040324	0.0000000	-3e-07
5	0.0481410	0.0481410	0.0481410	0.0000000	0e+00
6	0.0475780	0.0475780	0.0475780	0.0000000	0e+00
7	0.0352963	0.0352963	0.0352963	0.0000000	0e+00
8	-0.0224822	-0.0224822	-0.0224822	0.0000000	1e-07
9	-0.0810816	-0.0810816	-0.0810816	0.0000000	0e+00
10	0.1000712	0.1000712	0.1000712	0.0000000	0e+00
11	-0.0214049	-0.0214049	-0.0214049	0.0000000	0e+00
12	0.1096164	0.1096164	0.1096164	0.0000000	0e+00
13	0.0413296	0.0413296	0.0413296	0.0000000	0e+00
14	0.0198349	0.0198349	0.0198349	0.0000000	0e+00
15	-0.0392760	-0.0392760	-0.0392760	0.0000000	0e+00
16	-0.0863399	-0.0863399	-0.0863399	0.0000000	0e+00
17	0.0559711	0.0559711	0.0559711	0.0000000	0e+00
18	-0.1308629	-0.1308629	-0.1308629	0.0000000	0e+00
19	0.0437516	0.0437516	0.0437516	0.0000000	0e+00
20	0.0165890	0.0165890	0.0165890	0.0000000	0e+00
21	0.0574167	0.0574167	0.0574167	0.0000000	0e+00
22	-0.0103233	-0.0103233	-0.0103233	0.0000000	0e+00
23	0.1729607	0.1729607	0.1729607	0.0000000	0e+00
24	0.0503555	0.0503555	0.0503555	0.0000000	0e+00
25	-0.0230129	-0.0230129	-0.0230129	0.0000000	0e+00
26	-0.0023300	-0.0023300	-0.0023300	0.0000000	0e+00

Variable	Jimmy	GLM	GLMNET	Difference	Change
27	0.0407837	0.0407837	0.0407837	0.0000000	0e+00
28	0.0106670	0.0106670	0.0106670	0.0000000	0e+00
29	-0.0713834	-0.0713834	-0.0713834	0.0000000	0e+00
30	0.0872577	0.0872577	0.0872577	0.0000000	0e+00

Test with Actual Data

```
set.seed(1)
epsilon <- 10^(-5)

n    <- nrow(data)
X    <- scale(data[ , -c(1, 2)])
#X    <- data[ , -c(1, 2)]
X    <- as.matrix(cbind(rep(1, n), X))
y    <- data$diagnosis

beta <- rep(0, ncol(X))
lambda_vec <- 1 - log(seq(exp(0.001), exp(1), 0.01))
lambda_vec <- exp(seq(log(0.4), log(0.0004), -0.1))
lambda_vec <- c(0.4, 0.38, 0.36)

for(lambda in lambda_vec) {
# (max(t(X) %*% y) / n)

for(outer in 1:3) {

# initialize parameters
#p <- map_dbl(logistic(X %*% beta), p_adj, epsilon)
#w <- map_dbl(p, w_adj, epsilon)
#z <- X %*% beta + (y - p) / w

terminate <- 0
iter <- 1
while(terminate < 1) {

  p <- map_dbl(logistic(X %*% beta), p_adj, epsilon)
  w <- map_dbl(p, w_adj, epsilon)
  z <- X %*% beta + (y - p) / w

  beta_old <- beta
  # initially go through all parameters
  K <- 1:ncol(X)
  #if(iter > 1) {
  #  K <- which(beta > 0)
  #}

  for(k in K) {
    x_k    <- X[ , k]
    x_notk <- X[ , -k]
    b_notk <- beta[-k]
```

```r
    # un-penalized coefficient update
    b_k_temp <- sum(w * (z - x_notk %*% b_notk) * x_k) / sum(w * x_k^2)
    # shrinkage update
    b_k      <- S(b_k_temp, lambda * (k > 1) / mean(w * x_k^2))

    # update beta vector along with other parameters
    beta[k] <- b_k
    #p <- map_dbl(logistic(X %*% beta), p_adj, epsilon)
    #w <- map_dbl(p, w_adj, epsilon)
    #z <- X %*% beta + (y - p) / w
  }

  iter <- iter + 1

  if(iter == 1000 | max(abs(beta - beta_old)) < 10^-12) {
    print(iter)
    terminate <- 1
  }

}

}

# True estimates from GLMNET
fit <- glmnet(X, y, family = "binomial", standardize = FALSE, lambda = lambda, thresh = 10^-12)

# results
results <- tibble(
    Variable   = 1:length(beta)
  , Name       = c("intercept", names(data[ , -c(1, 2)]))
  , Jimmy      = beta
  , GLMNET     = as.vector(fit$beta[ , ncol(fit$beta)])
  , Difference = abs(Jimmy - GLMNET)
) %>%
  filter(Jimmy != 0 | GLMNET != 0)

print(paste0("lambda = ", lambda))
print(results %>% knitr::kable())
}
```

```
## [1] 6
## [1] 2
## [1] 2
## [1] "lambda = 0.4"
##
##
## | Variable|Name      |      Jimmy| GLMNET| Difference|
## |--------:|:---------|----------:|------:|----------:|
## |        1|intercept | -0.5211495|      0|  0.5211495|
## [1] 5
## [1] 2
## [1] 2
## [1] "lambda = 0.38"
```

```
##
##
## | Variable|Name                 |      Jimmy|    GLMNET| Difference|
## |--------:|:--------------------|----------:|---------:|----------:|
## |        1|intercept            | -0.5211755| 0.0000000|  0.5211755|
## |       29|concave points_worst |  0.0143262| 0.0143262|  0.0000000|
## [1] 11
## [1] 2
## [1] 2
## [1] "lambda = 0.36"
##
##
## | Variable|Name                 |      Jimmy|    GLMNET| Difference|
## |--------:|:--------------------|----------:|---------:|----------:|
## |        1|intercept            | -0.5223775| 0.0000000|  0.5223775|
## |       29|concave points_worst |  0.0996217| 0.0996217|  0.0000000|
```

Second Attempt with Actual Data

```
logistic_lasso <- function(
  # a numeric design matrix or data frame with named columns
    inputs
  # a vector of outputs; we must have length(output) == nrow(inputs)
  , output
  # a vector of descending penalization factors, ideally on a logarithmic scale
  , lambda_vec
  # standardize inputs using scale
  , standardize   = TRUE
  # a buffer to prevent divergence when fitted probabilities approach 0 or 1
  , epsilon       = 10^-8
  # maximum number of updates to quadratic approximation of likelihood
  , outer_maxiter =  100
  # maximum number of cycles for coordinate descent given quadratic approximation
  , inner_maxiter = 1000
  # tolerance for convergence of coordinate descent
  , tolerance     = 10^-12
) {

  # standardize data unless otherwise specified
  if(standardize) {

    # format data
    X <- as.matrix(cbind(rep(1, nrow(inputs)), scale(inputs)))
    y <- output

  } else {

    # format data
    X <- as.matrix(cbind(rep(1, nrow(inputs)), inputs))
    y <- output

  }
```

```r
# initialize coefficients at origin
beta    <- rep(0, ncol(X))
beta_df <- NULL

# begin lambda decrement
for(lambda in lambda_vec) {

  outer_term <- 0
  outer_iter <- 1

  # update quadratic approximation, execute coordinate descent until convergence, repeat
  while(outer_term < 1) {

    # update quadratic approximation; i.e., taylor expand around current estimates
    p <- map_dbl(logistic(X %*% beta), p_adj, epsilon)
    w <- map_dbl(p, w_adj, epsilon)
    z <- X %*% beta + (y - p) / w

    inner_term <- 0
    inner_iter <- 1

    # given current quadratic approximation, execute coordinate descent
    while(inner_term < 1) {

      beta_old <- beta

      # execute a complete cycle of coordinate descent
      for(k in 1:ncol(X)) {

        # un-penalized coefficient update
        b_k_temp <- sum(w * (z - X[ , -k] %*% beta[-k]) * X[ , k]) / sum(w * X[ , k]^2)
        # shrinkage update
        b_k      <- S(b_k_temp, (k > 1) * lambda / mean(w * X[ , k]^2))
        # update coefficient vector
        beta[k] <- b_k

      }

      inner_iter <- inner_iter + 1

      if(inner_iter == inner_maxiter | max(abs(beta - beta_old)) < tolerance) {

        inner_term <- 1

      }

    }

    outer_iter <- outer_iter + 1

    if(outer_iter == outer_maxiter | inner_iter == 2) {

      outer_term <- 1
```

```r
      }

    }

    beta_df <- rbind(beta_df, t(beta))

  }

  # format data frame of coefficient estimates
  colnames(beta_df) <- c("intercept", names(inputs))
  beta_df <- as_tibble(beta_df)

  # extract number of variables selected for each lambda
  selected_vec <- apply(beta_df, 1, function(x) sum(x != 0) - 1)

  # output results
  list(lambda = lambda_vec, beta = beta_df, selected = selected_vec)

}
```

```r
bad_vars <- c(
    "area_mean", "area_worst", "perimeter_mean", "perimeter_worst", "radius_mean"
  , "perimeter_se", "area_se"
  , "concave points_worst", "concavity_mean"
  , "texture_worst"
)
my_inputs <- data %>% select(-bad_vars) %>% select(-c(1, 2))
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(bad_vars)` instead of `bad_vars` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
my_output <- data$diagnosis

# identify minimum lambda value for which all coefficients are zero
lambda_max <- max(t(scale(as.matrix(my_inputs))) %*% y) / nrow(my_inputs)
# set lambda_min based on scaled data
lambda_min <- 0.0001
# define vector of lambdas
lambda_seq <- exp(seq(log(lambda_max), log(lambda_min), length.out = 100))

output <- logistic_lasso(my_inputs, my_output, lambda_seq, standardize = TRUE)
output
```

```
## $lambda
##   [1] 0.3751568949 0.3452309884 0.3176922429 0.2923502367 0.2690297381
##   [6] 0.2475694934 0.2278211118 0.2096480398 0.1929246163 0.1775352043
##  [11] 0.1633733909 0.1503412519 0.1383486741 0.1273127328 0.1171571179
##  [16] 0.1078116068 0.0992115781 0.0912975654 0.0840148460 0.0773130622
##  [21] 0.0711458732 0.0654706350 0.0602481051 0.0554421714 0.0510196024
##  [26] 0.0469498176 0.0432046756 0.0397582799 0.0365868000 0.0336683060
##  [31] 0.0309826174 0.0285111637 0.0262368555 0.0241439666 0.0222180255
##  [36] 0.0204457149 0.0188147798 0.0173139428 0.0159328261 0.0146618798
```

```
##  [41] 0.0134923157 0.0124160466 0.0114256305 0.0105142188 0.0096755097
##  [46] 0.0089037035 0.0081934636 0.0075398789 0.0069384299 0.0063849580
##  [51] 0.0058756360 0.0054069422 0.0049756356 0.0045787339 0.0042134927
##  [56] 0.0038773864 0.0035680910 0.0032834678 0.0030215487 0.0027805227
##  [61] 0.0025587230 0.0023546161 0.0021667907 0.0019939478 0.0018348925
##  [66] 0.0016885249 0.0015538329 0.0014298851 0.0013158246 0.0012108625
##  [71] 0.0011142732 0.0010253887 0.0009435944 0.0008683248 0.0007990593
##  [76] 0.0007353191 0.0006766634 0.0006226866 0.0005730155 0.0005273066
##  [81] 0.0004852439 0.0004465364 0.0004109166 0.0003781382 0.0003479745
##  [86] 0.0003202169 0.0002946735 0.0002711677 0.0002495369 0.0002296316
##  [91] 0.0002113141 0.0001944578 0.0001789461 0.0001646717 0.0001515360
##  [96] 0.0001394481 0.0001283245 0.0001180882 0.0001086684 0.0001000000
##
## $beta
## # A tibble: 100 x 21
##    intercept texture_mean smoothness_mean compactness_mean `concave points_mean`
##        <dbl>        <dbl>           <dbl>            <dbl>                 <dbl>
##  1    -0.521            0               0                0              2.22e-16
##  2    -0.523            0               0                0              6.99e- 2
##  3    -0.528            0               0                0              1.33e- 1
##  4    -0.534            0               0                0              1.92e- 1
##  5    -0.541            0               0                0              2.49e- 1
##  6    -0.547            0               0                0              3.05e- 1
##  7    -0.554            0               0                0              3.60e- 1
##  8    -0.560            0               0                0              4.14e- 1
##  9    -0.565            0               0                0              4.69e- 1
## 10    -0.570            0               0                0              5.23e- 1
## # ... with 90 more rows, and 16 more variables: symmetry_mean <dbl>,
## #   fractal_dimension_mean <dbl>, radius_se <dbl>, texture_se <dbl>,
## #   smoothness_se <dbl>, compactness_se <dbl>, concavity_se <dbl>,
## #   concave points_se <dbl>, symmetry_se <dbl>, fractal_dimension_se <dbl>,
## #   radius_worst <dbl>, smoothness_worst <dbl>, compactness_worst <dbl>,
## #   concavity_worst <dbl>, symmetry_worst <dbl>, fractal_dimension_worst <dbl>
##
## $selected
##   [1]  1  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3  3  4  6  6  6  6  6  6
##  [26]  6  6  6  6  6  6  6  6  6  6  6  7  7  7  7  7  7  8  8  8  8  8  8  8  9
##  [51]  9  9  9 10 10 10 10 10 11 11 11 12 13 15 15 15 15 15 14 14 16 16 16 16 16
##  [76] 16 16 16 16 16 17 17 18 18 17 17 17 18 18 18 19 19 19 19 20 20 20 20 20 20
```