Breast Cancer Diagnosis

Data import & cleanness

```
# Import data
breast_cancer =
  read_csv("./data/breast-cancer.csv") %>%
  select(-c(1, 33)) %>%
  mutate(diagnosis =
           as.numeric(as.factor(recode(diagnosis, `M` = 1, `B` = 0))) - 1) %>%
  mutate_each_(funs(scale(.)), c(2:31))

# Standardize predictors
# Newton-Raphson Algorithm
pred_1 <- as.tibble(breast_cancer[2:11])
bc_scale1 <- as.matrix(cbind(rep(1, nrow(breast_cancer)), pred_1, breast_cancer$diagnosis))
names(bc_scale1) = c("intercept", names(breast_cancer)[2:11], "outcome")
bc_scale1_x <- bc_scale1[, -12]
bc_scale1_y <- bc_scale1[, 12]

# Coordinate-wise descending Algorithm
pred_2 <- as.tibble(breast_cancer[2:31])
bc_scale2 <- as.matrix(cbind(rep(1, nrow(breast_cancer)), pred_2, breast_cancer$diagnosis))
names(bc_scale1) = c("intercept", names(breast_cancer)[2:11], "outcome")
bc_scale2_x <- bc_scale2[, -32]
bc_scale2_y <- bc_scale2[, 32]
```

Modified Newton-Raphson method

```
# Compute likelihood function, gradient, and Hessian matrix
compute_stat <- function(dat, betavec){
  x <- dat
  y <- breast_cancer$diagnosis
  u <- x %*% betavec
  pi <- exp(u) / (1 + exp(u))
  # Log-likelihood at betavec
  loglik <- sum(y * u - log(1 + exp(u)))
  # Gradient at betavec
  grad <- t(x) %*% (y - pi)
  # Hessian at betavec
  hess <- -t(x) %*% diag(as.vector(pi * (1 - pi))) %*% x

  return(list(loglik = loglik, grad = grad, hess = hess))
}

# Modified Newton-Raphson method
NewtonRaphson <- function(dat, func, start, tol = 1e-15, maxier = 200) {
  i <- 0
```

```
  cur <- start

  # Computate log-likelihood, gradient, and Hessian matrix
  stuff <- func(dat, cur)
  res <- c(0, stuff$loglik, cur)
  prevloglik <- -Inf

  while (i < maxier && abs(stuff$loglik - prevloglik) > tol) {
    i <- i + 1
    step <- 1
    prevloglik <- stuff$loglik
    prev <- cur
    cur <- prev - solve(stuff$hess) %*% stuff$grad

    # Step halving and re-direction
    while (func(dat, cur)$loglik < stuff$loglik) {
      if (!all(eigen(stuff$hess)$values < 0)) {
        gamma <- max(eigen(stuff$hess)$values)
        new_hess <- stuff$hess - gamma * diag(nrow = dim(dat)[1])
        cur <- prev - solve(new_hess) %*% stuff$grad
      }
      else {
        step <- step / 2
        cur <- prev - step * solve(stuff$hess) %*% stuff$grad
      }
    }
    stuff <- func(dat, cur)

    res <- rbind(res, c(i, stuff$loglik, cur))
  }

  return(res)
}
```

NR results

```
# NewtonRaphson(bc_scale1_x, compute_stat, rep(1, 11))
# NewtonRaphson(bc_scale1_x, compute_stat, rep(10, 11))
```

Test `glm`

```
glm(diagnosis ~., family = binomial(link = "logit"), data = breast_cancer[, 1:31])
```

```
##
## Call:  glm(formula = diagnosis ~ ., family = binomial(link = "logit"),
##     data = breast_cancer[, 1:31])
##
## Coefficients:
##             (Intercept)             radius_mean            texture_mean
##                  720.23               -9421.17                  217.20
##          perimeter_mean               area_mean         smoothness_mean
##                 2120.94                6947.25                  286.66
##        compactness_mean          concavity_mean    `concave points_mean`
```

```
##                -1357.19                  1123.49                   536.85
##           symmetry_mean   fractal_dimension_mean                radius_se
##                -261.61                   221.88                   452.01
##              texture_se             perimeter_se                 area_se
##                -114.62                -1016.24                  2148.07
##           smoothness_se           compactness_se             concavity_se
##                -157.90                   775.78                 -1055.02
##     `concave points_se`              symmetry_se     fractal_dimension_se
##                 870.40                  -377.81                  -868.27
##            radius_worst           texture_worst         perimeter_worst
##                3432.55                   220.93                   721.55
##              area_worst         smoothness_worst        compactness_worst
##               -2653.87                   -54.74                  -517.93
##         concavity_worst   `concave points_worst`           symmetry_worst
##                 539.23                   174.16                   598.34
## fractal_dimension_worst
##                 454.68
##
## Degrees of Freedom: 567 Total (i.e. Null);  537 Residual
## Null Deviance:       750.5
## Residual Deviance: 0.0001214     AIC: 62
```

Logistic-LASSO model

```r
# Soft Threshold function
sf <- function(beta, lambda) {
  if (beta > 0 & lambda < abs(beta)) {
    beta <- beta - lambda
  }
  else if (beta < 0 & lambda < abs(beta)) {
    beta <- beta + lambda
  }
  else{
    beta <- 0
  }
  return(beta)
}


# Coordinate-wise LASSO
cd_lasso <- function(dat, betavec, lambda, maxier = 2000, tol = 1e-15) {
  x <- dat
  y <- breast_cancer$diagnosis
  i <- 0
  loglik <- 0
  prevloglik <- -Inf
  res <- c(0, loglik, betavec)

  while (i < maxier & abs(loglik - prevloglik) > tol) {
    i <- i + 1

    for (j in 1:length(betavec)) {
      u <- x %*% betavec
      pi <- exp(u) / (1 + exp(u))
```

```
      w <- pi * (1 - pi)
      w <- ifelse(abs(w)<1e-5, 1e-5, w)  # set up a lower bound o.w. omega will introduce NaN
      z <- x %*% betavec + (y - pi)/w
      z_dej <- x[, -j] %*% betavec[-j]
      betavec[j] <-
        sf(sum(w * x[, j] * (z - z_dej)), lambda) / (sum(w * x[, j]^2))
    }

    loglik <- sum(w * (z - x %*% betavec)^2) / (2 * dim(x)[1]) + lambda * sum(abs(betavec))

    prevloglik <- loglik
    res <- rbind(res, c(i, loglik, betavec))
  }

  return(res)
}
cd_lasso(bc_scale2_x, rep(1,31), lambda = 0.8)
```

```
##     [,1]     [,2]      [,3]     [,4]     [,5]    [,6]     [,7]     [,8]
## res    0   0.0000  1.000000 1.000000 1.000000 1.00000 1.000000 1.000000
##        1 209.9322 -1.455046 8.933849 7.012158 5.40157 3.995846 2.197537
##          [,9]    [,10]    [,11]    [,12]      [,13]     [,14]     [,15]
## res  1.000000 1.000000 1.000000 1.000000 1.00000000  1.000000  1.000000
##     -8.008097 1.768655 4.131473 2.454116 0.01965311 -4.890822 -1.612933
##        [,16]    [,17]     [,18]      [,19]     [,20]   [,21]     [,22]
## res 1.000000 1.000000  1.000000  1.0000000 1.0000000 1.00000 1.0000000
##     3.096116 1.895223 -0.600698 -0.1117062 0.2876601 1.77098 0.4986397
##         [,23]      [,24] [,25]      [,26] [,27]    [,28] [,29]    [,30]
## res 1.0000000  1.0000000     1  1.0000000     1 1.000000     1 1.000000
##     0.8570578 -0.1050368     0 -0.1892118     0 4.282736     0 1.224503
##         [,31]    [,32]     [,33]
## res 1.0000000 1.000000 1.0000000
##     0.3240427 1.253133 0.6491268
```

5-fold CV