HW 2

Bryan Mui - UID 506021334 - 28 April 2025

Loaded packages: ggplot2, tidyverse (include = false for this chunk)

Reading the dataset:

```r
data <- read_csv("dataset-logistic-regression.csv")
```

```
Rows: 10000 Columns: 101
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (101): y, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(data, n = 25)
```

```
# A tibble: 25 x 101
       y      X1      X2     X3     X4     X5     X6      X7     X8      X9
   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl>   <dbl>
 1     1 -0.0895  0.450   1.71   0.657 -0.392  1.24   0.895   1.13  -0.0117
 2     1 -0.0943  0.281  -0.147 -0.701  0.400 -0.210  0.677  -0.440   0.458
 3     0 -0.431  -0.445  -0.777 -0.832 -2.26  -1.62  -1.98   -1.67   -1.15
 4     0  0.644   0.0817 -0.448  0.852 -1.02   0.671  0.299   0.145 -0.205
 5     1 -0.919  -0.0241  0.807 -0.612 -0.498  0.350  1.12    0.242 -0.947
 6     0 -1.89   -1.11   -0.210  0.161 -1.34  -2.04  -0.0135 -1.39  -1.31
 7     0 -1.34   -0.804   0.322 -0.110  0.624 -0.329 -0.432  -0.191  0.171
 8     1  0.329   0.468   0.719  0.588  1.71   1.39   0.603   0.650  0.161
 9     0  0.332   1.42   -0.431  1.02   0.484  0.348  0.474   1.26  -0.479
10     0 -0.311   0.0193  0.168 -0.346  0.626 -0.704 -0.290   0.680 -0.0453
# i 15 more rows
```

```
# i 91 more variables: X10 <dbl>, X11 <dbl>, X12 <dbl>, X13 <dbl>, X14 <dbl>,
#   X15 <dbl>, X16 <dbl>, X17 <dbl>, X18 <dbl>, X19 <dbl>, X20 <dbl>,
#   X21 <dbl>, X22 <dbl>, X23 <dbl>, X24 <dbl>, X25 <dbl>, X26 <dbl>,
#   X27 <dbl>, X28 <dbl>, X29 <dbl>, X30 <dbl>, X31 <dbl>, X32 <dbl>,
#   X33 <dbl>, X34 <dbl>, X35 <dbl>, X36 <dbl>, X37 <dbl>, X38 <dbl>,
#   X39 <dbl>, X40 <dbl>, X41 <dbl>, X42 <dbl>, X43 <dbl>, X44 <dbl>, ...
```

Our data set has 10000 observations, 1 binary outcome variable y, and 100 predictor variables X1-X100

Separating into X matrix and y vector:

```
X <- data %>%
  select(-y)
y <- data %>%
  select(y)
```

Problem 1

Part (α)

The optimization problem is to minimize the log-likelihood function. From there we will get the objective function and gradient function

From the slides in class we have:

$$\min_{\beta}(-\ell(\beta)) = \frac{1}{m}\sum_{i=1}^{m} f_i(\beta)$$

and the equation for $f_i(\beta)$:

$$f_i(\beta) = -y_i(x_i^\intercal\beta) + log(1 + exp(x_i^\intercal\beta))$$

For the objective function, we get:

$$f(\beta) = \frac{1}{m}\sum_{i=1}^{m}[-y_i(x_i^\intercal\beta) + log(1 + exp(x_i^\intercal\beta))]$$

We can matricize the objective function to

$$\boxed{f(\beta) = \frac{1}{m}[-y^\intercal(X\beta) + \mathbf{1}^\intercal log(1 + exp(X\beta))]}$$

We also have the gradient function:

$$\nabla f(x) = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(x)$$

and

$$\nabla_\beta f_i(\beta) = [\sigma(x_i^\intercal \beta) - y_i] \cdot x_i$$

where $\sigma(z) = \frac{1}{1+exp(-z)}$ as the logistic sigmoid function, therefore:

$$\nabla f(x) = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(x),\ \nabla_\beta f_i(\beta) = [\sigma(x_i^\intercal \beta) - y_i] \cdot x_i$$
$$\nabla f(\beta) = \frac{1}{m}\sum_{i=1}^{m}[\sigma(x_i^\intercal \beta) - y_i] \cdot x_i$$

And we can also matricize this:

$$\boxed{\nabla f(\beta) = \frac{1}{m}X^\intercal[\sigma(X\beta) - y], \quad \sigma(z) = \frac{1}{1 + exp(-z)}}$$

Therefore our gradient descent update step is(for constant step size):

$$\boxed{\beta_{k+1} = \beta_k - \eta \nabla f(\beta_k)}$$

Implement the following algorithms to obtain estimates of the regression coefficients $\boldsymbol{\beta}$:

(1) Gradient descent with backtracking line search

Algorithm; Backtracking Line Search:

Params:

- Set $\eta^0 > 0$(usually a large value ~1),
- Set $\eta_1 = \eta^0$
- Set $\epsilon \in (0, 1), \tau \in (0, 1)$, where ϵ and τ are used to modify step size

Repeat:

- At iteration k, set $\eta_k < -\eta_{k-1}$

  1. Check whether the Armijo Condition holds:

  $$h(\eta_k) \leq h(0) + \epsilon \eta_k h'(0)$$

  where $h(\eta_k) = f(x_k) - \eta_k \nabla f(x_k)$,
  and $h(0) = f(x_k)$,
  and $h'(0) = -||\nabla(x_k)||^2$

  2.

  - If yes(condition holds), terminate and keep η_k
  - If no, set $\eta_k = \tau \eta_k$ and go to Step 1

Stopping criteria: Stop if $||x_k - x_{k+1}|| \leq tol$ (change in parameters is small)

Implement BLS

```
# logistic gradient descent w/ bls
log_bls <- function(X, y, tol = 1e-6, max_iter = 10000, epsilon = 0.5, tau =
↪  0.5) {
  # Initialize
  n <- nrow(X)
  p <- ncol(X)
  x <- as.matrix(X)
  y <- as.matrix(y)
  beta <- as.matrix(rep(0, p))
  obj_values <- numeric(max_iter)
  eta_values <- numeric(max_iter)  # To store eta values used each iteration
  beta_values <- list() # To store beta values used each iteration
  eta_bt <- 1  # Initial step size for backtracking

  # Objective function: negative log-likelihood
```

```
# input: Beta vector, x matrix, y matrix
# output: scalar objective func value
# comments: We want to minimize this function for logit regression
obj_function <- function(beta, x, y) {
  m <- nrow(x)
  z <- x %*% beta
  (1 / m) * (-(t(y) %*% z) + sum(log(1 + exp(z))))
}

# Gradient function
# input: Beta vector, x matrix, y matrix
# output: gradient vector in the dimension of nrow(Beta) x 1
# comments: We use this for gradient descent
gradient <- function(beta, x, y) {
  m <- nrow(x)                          # define m
  sig <- function(z) 1 / (1 + exp(-z))  # sigmoid function
  (1 / m) * (t(x) %*% (sig(x %*% beta) - y))
}

# Algorithm:
for (iter in 1:max_iter) {
  grad <- gradient(beta, x, y)

  #cat("iter ", iter, "\n")

  # backtracking step
  current_obj <- obj_function(beta, x, y)
  grad_norm_sq <- sum(grad^2)

  beta_new <- beta - eta_bt * grad

  while (obj_function(beta_new, x, y) > current_obj - epsilon * eta_bt *
   ↪  grad_norm_sq) {
    eta_bt <- tau * eta_bt
    beta_new <- beta - eta_bt * grad
  }

  # save values to the matrix
  eta_values[iter] <- eta_bt
  obj_values[iter] <- obj_function(beta_new, x, y)
  beta_values[[iter]] <- beta_new

  if (sqrt(sum((beta_new - beta)^2)) < tol) {
```

```r
      # set the vector ranges and break
      beta <- beta_new
      obj_values <- obj_values[1:iter]
      eta_values <- eta_values[1:iter]
      beta_values <- beta_values[1:iter]
      break
    }

    beta <- beta_new
  }

  return(list(beta = beta, obj_values = obj_values, eta_values = eta_values,
   ↪  beta_values = beta_values))
}
```

TESTING: BLS

```r
log_reg_bls <- log_bls(X, y, tol=1e-6, max_iter=10000, epsilon=0.5, tau=0.5)
```

```r
cat("betas \n")
```

```
betas
```

```r
print(log_reg_bls$beta)
```

```
                y
X1   -0.1418188273
X2   -0.0601340162
X3    0.1588169528
X4    0.1328223189
X5   -0.0480437781
X6    0.0992481092
X7    0.1189707785
X8    0.1165560855
X9    0.0121222291
X10   0.0002641372
X11   0.0440526577
X12  -0.1793886158
X13  -0.0107332284
```

```
X14  -0.1230510680
X15   0.0724799230
X16   0.0571868940
X17   0.1299458439
X18   0.1249113906
X19  -0.0018170795
X20   0.1248825007
X21  -0.0107845610
X22  -0.1431801553
X23  -0.1094846603
X24   0.0576435159
X25  -0.1190174922
X26   0.0164879978
X27  -0.0977482724
X28   0.1544632196
X29  -0.0276524076
X30   0.0164226883
X31  -0.0589010945
X32   0.0205242099
X33   0.1352153619
X34  -0.0301792708
X35  -0.0097106467
X36   0.0631274232
X37   0.1972595891
X38   0.0932479560
X39   0.1242393813
X40   0.1466042152
X41   0.1112967707
X42  -0.1226544766
X43  -0.0374866338
X44  -0.0155583465
X45  -0.0103256878
X46  -0.1807311531
X47   0.0122916067
X48   0.0309436582
X49   0.0257891274
X50   0.1230837280
X51  -0.0237134869
X52  -0.0136672407
X53   0.0802510780
X54   0.1695795679
X55   0.1711403640
X56  -0.0447703054
```

```
X57  -0.0407325139
X58  -0.0768578382
X59   0.0786448045
X60  -0.1192193182
X61  -0.0080431756
X62   0.0701535429
X63   0.0295238798
X64  -0.1090225592
X65   0.0633967271
X66  -0.1450871355
X67   0.1404424947
X68   0.0649021774
X69  -0.1595801011
X70   0.1128079446
X71   0.1888668197
X72   0.0920649207
X73  -0.0647758044
X74  -0.0684344716
X75   0.2306707321
X76  -0.1312078759
X77   0.0301767178
X78  -0.0742090881
X79   0.0695790861
X80  -0.0273839196
X81   0.0183730389
X82   0.0555339156
X83  -0.0196159895
X84  -0.0119020076
X85   0.0981161430
X86   0.1724354285
X87   0.0832570899
X88  -0.0070115810
X89   0.0720539875
X90   0.0779093972
X91   0.0026928031
X92  -0.1223692130
X93   0.0073627318
X94  -0.0996425700
X95  -0.0485788118
X96   0.0338587696
X97   0.1496954257
X98   0.1702285222
X99   0.0197714549
```

```
X100  0.0070161693
```

```r
cat("The function converged after", length(log_reg_bls$obj_values), "
 ↪  iterations \n")
```

```
The function converged after 1909  iterations
```

```r
cat("Eta Vals: \n")
```

```
Eta Vals:
```

```r
print(log_reg_bls$eta_values[1:50])
```

```
 [1] 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625
[11] 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625
[21] 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625
[31] 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625
[41] 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625 0.0625
```

```r
cat("Objective Function vals \n")
```

```
Objective Function vals
```

```r
print(log_reg_bls$obj_values[1:50])
```

```
 [1] 0.5642463 0.5491551 0.5446388 0.5427888 0.5418291 0.5412092 0.5407301
 [8] 0.5403132 0.5399257 0.5395537 0.5391908 0.5388343 0.5384829 0.5381361
[15] 0.5377934 0.5374547 0.5371200 0.5367892 0.5364622 0.5361389 0.5358194
[22] 0.5355035 0.5351913 0.5348826 0.5345775 0.5342759 0.5339777 0.5336830
[29] 0.5333916 0.5331036 0.5328188 0.5325373 0.5322591 0.5319840 0.5317120
[36] 0.5314432 0.5311774 0.5309146 0.5306549 0.5303981 0.5301442 0.5298932
[43] 0.5296451 0.5293997 0.5291572 0.5289174 0.5286804 0.5284460 0.5282142
[50] 0.5279851
```

(2) Gradient descent with backtracking line search and Nesterov momentum

Nesterov is simply BLS with a special way to select the momentum ξ,

We set ξ to:

$$\frac{k-1}{k+2}$$

where k is the iteration index

Algorithm(Nesterov Momentum with BLS)

Params:

- Set $\eta^0 > 0$(usually a large value ~1),
- Set $\eta_1 = \eta^0$
- Set $\epsilon \in (0,1), \tau \in (0,1)$, where ϵ and τ are used to modify step size

Repeat:

- At iteration k, set $\eta_k < -\eta_{k-1}$, update with

$$\boxed{x_{k+1} = y_k - \eta_k \nabla(f(y_k)), \quad y_k = x_k + \xi(x_k - x_{k-1}), \quad \xi = \frac{k-1}{k+2}}$$

- Check the next setting of η:

  1. Check whether the Armijo Condition holds:

$$h(\eta_k) \leq h(0) + \epsilon \eta_k h'(0)$$

  where $h(\eta_k) = f(x_k) - \eta_k \nabla f(x_k)$,
  and $h(0) = f(x_k)$,
  and $h'(0) = -||\nabla(x_k)||^2$

  2.

  - If yes(condition holds), terminate and keep η_k
  - If no, set $\eta_k = \tau \eta_k$ and go to Step 1

Stopping criteria: Stop if $||x_k - x_{k+1}|| \leq tol$ (change in parameters is small)

(3) Gradient descent with AMSGrad-ADAM momentum

(no backtracking line search, since AMSGrad-ADAM adjusts step sizes per parameter using momentum and adaptive scaling)

(4) Stochastic gradient descent with a fixed schedule of decreasing step sizes

(5) Stochastic gradient descent with AMSGrad-ADAM-W momentum

Part (a)

Part (b)