Problem Set #5

Ian Bach

2024-09-21

1A).

Compute the Gradient of Log-Likelihood log

$$L(\beta \mid y)$$

:

$$\log L(\beta \mid y) = \sum_{i=1}^{n} \left(y_i \log(p(x_i;\beta)) + (1-y_i)\log(1-p(x_i;\beta))\right)$$

The gradient of the log-likelihood with respect to

$$\beta_j \quad \text{for } j = 1, 2, \dots, k$$

For the intercept B0 is

$$\frac{\partial \log L(\beta \mid y)}{\partial \beta_j} = \sum_{i=1}^{n} \left(y_i - p(x_i;\beta)\right) x_{ij}$$

The

$$-\lambda \sum_{j=1}^{k} \beta_j^2$$

The gradient of the penalty term with respect to each j is

$$\frac{\partial}{\partial \beta_j}\left(-\lambda \sum_{j=1}^{k} \beta_j^2\right) = -2\lambda\beta_j$$

We are able to cobmine the gradient of the log-likelihoo dand the gradient of the pentaly term to obtain the graident of the penalized log-likelihood.

For

$$\beta_j \quad \text{for } j = 1, 2, \dots, k$$

$$\frac{\partial f(\beta)}{\partial \beta_j} = \sum_{i=1}^{n} \left(y_i - p(x_i;\beta)\right) x_{ij} - 2\lambda\beta_j$$

For the Intercept 0 due to its not penalized:

$$\frac{\partial f(\beta)}{\partial \beta_0} = \sum_{i=1}^{n} \left(y_i - p(x_i;\beta)\right)$$

The gradient of the penalized log-likelihood is:

$$\nabla f(\beta) = \begin{cases} \sum_{i=1}^{n}(y_i - p(x_i;\beta)), & \text{for } \beta_0 \\ \sum_{i=1}^{n}(y_i - p(x_i;\beta))x_{ij} - 2\lambda\beta_j, & \text{for } \beta_j \ (j = 1, 2, \dots, k) \end{cases}$$

1B).

```r
library(readr)
setwd("C:/Users/ibach/OneDrive - Terillium/Desktop/Purdue MSBA/Machine Learning/HW 5")

# Load the dataset
data <- read_csv("logit_ridge.csv", col_names = FALSE)
```

```
## Rows: 100 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (21): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, ...
##
## 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
# Split data into predictors (X) and response (y)
X <- as.matrix(data[, -1])
y <- as.vector(data[[1]])

# Split data into training and test sets
X_train <- X[11:100, ]
y_train <- y[11:100]
X_test <- X[1:10, ]
y_test <- y[1:10]

# Define necessary parameters
lambda <- 1     # Ridge regularization parameter
alpha <- 1e-4   # Learning rate
iterations <- 10000   # Number of iterations

# Initialize beta (coefficients)
beta <- rep(0, ncol(X_train))

# Sigmoid function
sigmoid <- function(z) {
  1 / (1 + exp(-z))
}

# Log-likelihood function with ridge penalty
log_likelihood_ridge <- function(X, y, beta, lambda) {
  linear_pred <- X %*% beta
  log_likelihood <- sum(y * linear_pred - log(1 + exp(linear_pred)))
  penalty <- lambda * sum(beta^2)
  return(log_likelihood - penalty / 2)
}
```

```r
# Gradient ascent function
gradient_ascent <- function(X, y, beta, alpha, lambda, iterations) {
  for (i in 1:iterations) {
    linear_pred <- X %*% beta
    prob <- sigmoid(linear_pred)
    gradient <- t(X) %*% (y - prob) - lambda * beta
    beta <- beta + alpha * gradient
  }
  return(beta)
}

# Apply gradient ascent to estimate beta
beta_estimates <- gradient_ascent(X_train, y_train, beta, alpha, lambda, iterations)

# Print the estimates for beta_1 and beta_2
beta_1 <- beta_estimates[1]
beta_2 <- beta_estimates[2]

cat("Beta_1 estimate:", beta_1, "\n")
```

```
## Beta_1 estimate: -2.550722
```

```r
cat("Beta_2 estimate:", beta_2, "\n")
```

```
## Beta_2 estimate: -0.3183075
```

1C).

The prediction error for each observation in the test set is defined as:

$$e_i^2 = \left(y_i - \hat{p}(x_i; \hat{\beta})\right)^2$$

```r
# Sigmoid function for prediction
sigmoid <- function(z) {
  1 / (1 + exp(-z))
}

# Compute predicted probabilities for the test set using the estimated beta
predicted_probabilities <- sigmoid(X_test %*% beta_estimates)

# Compute squared errors for the test set
squared_errors <- (y_test - predicted_probabilities)^2

# Compute the average test error
average_test_error <- mean(squared_errors)

# Print the result
cat("Average test error:", average_test_error, "\n")
```

```
## Average test error: 0.0845192
```

2A).

The likelihood for a single observation yi is:

$$p(y_i \mid X, \beta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2\right)$$

For n independent observations, the likelihood for the entire dataset is:

$$L(\beta; X, y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2\right)$$

The log-likelihood is:

$$\log L(\beta; X, y) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2$$

2B).

The prior for 1,..., p is independent and identically distributed as double exponential (Laplace) with:

$$p(\beta) = \prod_{j=1}^{p} \frac{2}{\tau} e^{-\tau|\beta_j|} = \left(\frac{2}{\tau}\right)^p \exp\left(-\frac{\tau}{1}\sum_{j=1}^{p}|\beta_j|\right)$$

Assuming a normal likelihood, the likelihood function is:

$$p(y \mid \beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^\top (y - X\beta)\right)$$

Bayes Theorem:

By Bayes' theorem, the posterior is proportional to the likelihood times the prior:

$$p(\beta \mid y) \propto p(y \mid \beta) p(\beta)$$

Substituting the expressions for the likelihood and prior:

$$p(\beta \mid y) \propto \exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^\top (y - X\beta) - \tau\sum_{j=1}^{p}|\beta_j|\right)$$

2C).

Posterior Distribution:

$$p(\beta \mid y) \propto \exp\left(-\frac{1}{2\sigma^2}\|y - X\beta\|_2^2 - \tau\sum_{j=1}^{p}|\beta_j|\right)$$

Maximizing p(y) is equivalent to minimizing the negative log-posterior:

$$-\log p(\beta \mid y) = \frac{1}{2\sigma^2}\|y - X\beta\|_2^2 + \tau \sum_{j=1}^{p} |\beta_j|$$

Lasso Objective:

$$\frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

By setting

$$\lambda = \tau_1$$

, the mode of the posterior distribution is identical to the Lasso estimate showing as both minimize the same function