Homework 3 (Oct. 30th)

Deadline: Wednesday, November 16th, at 11:59pm.

Submission: Read the submission instruction carefully! There are 4 questions in this assignment. You need to submit two files through Quercus for this assignment.

- The first file should be a PDF file titled `hw3_writeup.pdf` containing your answers to Questions 1 – 4, as well as R code and R outputs requested for Questions 3 and 4. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.
- The second file should be your completed R code, named as `penalized_logistic_regression.R`. You need to ensure that this file has the exact name as indicated. DO NOT set or modify the working directory within this file.

Neatness Point: You will be deducted one point if we have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the total possible marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

- **Problem 1 (3 pts)**
  Consider the classification problem with the label of Y belong to $\mathcal{C} := \{1, 2, \ldots, K\}$ and any realization x of $X \in \mathbb{R}^p$. Let f be any classifier that maps any $x \in \mathbb{R}^p$ to a label in $\mathcal{C}$.

  1. (**2 pts**) Prove that the best function f^* (i.e. the Bayes classifier)

  $$f^* := \underset{f:\mathbb{R}^p \to \mathcal{C}}{\operatorname{argmin}} \ \mathbb{E}\Big[1\{Y \neq f(X)\} \mid X = x\Big]$$

  satisfies

  $$f^*(x) = \underset{k \in \mathcal{C}}{\operatorname{argmax}} \ \mathbb{P}(Y = k \mid X = x). \tag{0.1}$$

  2. (**1 pt**) Argue that the Bayes error equals to

  $$\mathbb{E}\Big[1\{Y \neq f^*(X)\} \mid X = x\Big] = 1 - \max_{k \in \mathcal{C}} \ \mathbb{P}(Y = k \mid X = x).$$

- **Problem 2 (3 pts)**

  Consider a classification problem. Assume that the response variable Y can only take value in $\mathcal{C} = \{1, 2, 3\}$. For a fixed x_0, assume that the conditional probability of Y given $X = x_0$ follows

  $$\mathbb{P}(Y = 1 \mid X = x_0) = 0.6; \quad \mathbb{P}(Y = 2 \mid X = x_0) = 0.3; \quad \mathbb{P}(Y = 3 \mid X = x_0) = 0.1.$$

  Consider a naive classifier $\hat{f}$, called random guessing, which randomly picks one label from $\mathcal{C} = \{1, 2, 3\}$ with equal probability.

  1. (**2 pts**) Compute the expected test error rate of $\hat{f}$ at $X = x_0$.

  2. (**1 pt**) Compute the Bayes error rate at $X = x_0$ and compare it with that of $\hat{f}$.

- **Problem 3 (21 pts)**

  In this problem, you will implement logistic regression by completing the provided code in `penalized_logistic_regression.R` & `hw3_starter.R` and experiment with the completed code.

  Throughout this homework, you will be working with a subset of hand-written digits, 2's and 3's, represented as 16×16 pixel arrays. We show the example digits in Figure 1. The pixel intensities are between 0 and 1, and were read into the vectors in a raster-scan manner. You are given one training set: `train` which contains 300 examples of each class. You can access and load this training set by using functions

```
source("hw3_starter/utils.R")
data_train <- Load_data("hw3_starter/data/train.csv")
x_train <- train$x
y_train <- train$y
```

  `y_train` contains the labels of these 300 images while `x_train` are the 256 pixel values. You are also given a validation set that you should use for tuning and a test set that you should use for reporting the final performance. Optionally, the code for visualizing the dataset is located at `utils.py`.

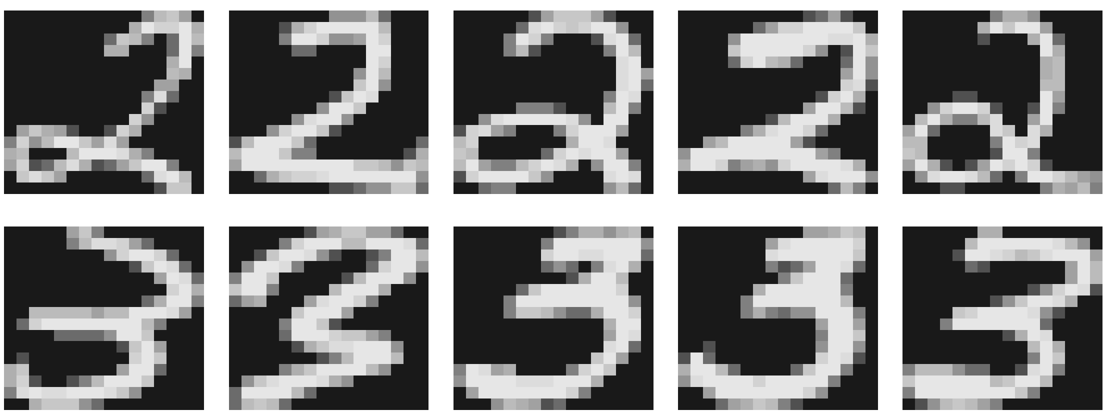


Figure 1: Example digits. Top and bottom show digits of 2s and 3s, respectively.

  You need to implement the penalized logistic regression model by minimizing the cost

$$\mathcal{J}(\boldsymbol{\beta}, \beta_0) \;:=\; -\frac{1}{n}\sum_{i=1}^{n}\Big\{y_i \log\big[p(\boldsymbol{x}_i; \boldsymbol{\beta}, \beta_0)\big] + (1-y_i)\log\big[1 - p(\boldsymbol{x}_i; \boldsymbol{\beta}, \beta_0)\big]\Big\} + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$$

  over $(\boldsymbol{\beta}, \beta_0) \in (\mathbb{R}^p, \mathbb{R})$, where

$$p(\boldsymbol{x}_i; \boldsymbol{\beta}, \beta_0) = \frac{e^{\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}}}{1 + e^{\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}}}.$$

  Here n is the total number of data points, p is the number of features in $\boldsymbol{x}_i$, $\lambda \geq 0$ is the regularization parameter and $\boldsymbol{\beta}$ and β_0 are the parameters to optimize over. Note that we should only penalize the coefficient parameters $\boldsymbol{\beta}$ and not the intercept term β_0.

1. (**2 pts**) Verify that the gradients of $\mathcal{J}(\boldsymbol{\beta}, \beta_0)$ at any $(\bar{\boldsymbol{\beta}}, \bar{\beta}_0)$ have the following expression,

$$\frac{\partial \mathcal{J}(\boldsymbol{\beta}, \beta_0)}{\partial \boldsymbol{\beta}}\Big|_{\bar{\boldsymbol{\beta}}, \bar{\beta}_0} = \frac{1}{n}\sum_{i=1}^{n}\left[-y_i + \frac{e^{\bar{\beta}_0 + \boldsymbol{x}_i^\top \bar{\boldsymbol{\beta}}}}{1 + e^{\bar{\beta}_0 + \boldsymbol{x}_i^\top \bar{\boldsymbol{\beta}}}}\right]\boldsymbol{x}_i + \lambda \bar{\boldsymbol{\beta}},$$
$$\frac{\partial \mathcal{J}(\boldsymbol{\beta}, \beta_0)}{\partial \beta_0}\Big|_{\bar{\boldsymbol{\beta}}, \bar{\beta}_0} = \frac{1}{n}\sum_{i=1}^{n}\left[-y_i + \frac{e^{\bar{\beta}_0 + \boldsymbol{x}_i^\top \bar{\boldsymbol{\beta}}}}{1 + e^{\bar{\beta}_0 + \boldsymbol{x}_i^\top \bar{\boldsymbol{\beta}}}}\right].$$

2. (**4 pts**) Implement the functions `Evaluate`, `Predict_logis`, `Comp_gradient` and `Comp_loss` located at `penalized_logistic_regression.R`. While implementing the functions, remember to vectorize the operations; you should not have any `for-loops` in these functions. Include your code in the report.

   Important note: carefully read the provided code in `penalized_logistic_regression.R`. *You should understand the code and its structure instead of using it as a black box!*

3. (**2 pts**) Complete the missing parts in function `Penalized_Logistic_Reg` located at `penalized_logistic_regression.R`. This function should train the penalized logistic regression model using gradient descent on given training set. You may use the implemented functions from step 2. Include your code in the report.

   For parts 2 and 3, your completed `penalized_logistic_regression.R` *should NOT import other* `R` *packages.*

4. (**4 pts**) Complete the part (a) in `hw3_starter.R`.

   In this part, you need to fix your regularization parameter, `lbd = 0`, and to experiment with the hyperparameters for `stepsize` (the learning rate) and `max_iter` (the number of iterations).

   [Hints: (1) You only need to use the training data for this part. (2) A too small learning rate takes longer to converge. (3) A too large learning rate is also problematic.]

   – In the write-up, report and briefly explain which hyperparameter settings you found worked the best.
   – For this choice of hyperparameters, generate and report a plot that shows how the training loss changes (iteration counter on x-axis and training loss on y-axis).
   – For this choice of hyperparameters, generate and report a plot for the training 0-1 error (iteration counter on x-axis and training error on y-axis).
   – Did the training 0-1 error have the same pattern as the training loss? Is your finding aligned with your expectation? State you reasoning.

5. (**7 pts**) Complete the part (b) in `hw3_starter.R`.

   Using the selected setting of hyperparameters (for learning rate and number of iteration) that you identified in step 4, fit the model by using $\lambda \in \{0, 0.01, 0.05, 0.1, 0.5, 1\}$.

   – (**1 pts**) Does your selected setting of hyperparameters guarantee convergence for all λ's? If not, re-identify hyperparameters for those λ's for which convergence is not guranteed. Report the hyperparameter setting(s) you used for each λ.
   – (**2 pts**) Generate and report one plot that shows how the training 0-1 error changes as you train with different values of λ.
   – (**2 pts**) Generate and report one plot that shows how the validation 0-1 error changes as you train with different values of λ.

- (**2 pts**) Comment on the effects of λ based on these two plots. Which is the best value of λ based on your experiment?

6. (**2 pts**) Complete the part (c) in `hw3_starter.R`.
   Fit the model by using the best value of λ identified in step 5 and report its test 0-1 error. Compare your test error with the model fitted by using `glmnet` with the same λ.

- **Problem 4 (10 pts)**
  In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set.

  1. (**1 pts**) Create a binary variable, **mpg01**, that contains a 1 if **mpg** contains a value above its median, and a 0 if **mpg** contains a value below its median. You can compute the median using the `median()` function.

     Split the data into a training set (70%) and a test set (30%). (Use `set.seed(0)` to ensure reproducibility.)

  2. (**2 pts**) Perform LDA on the training data in order to classify **mpg01** using the variables **cylinders**, **displacement**, **horsepower**, **weight**, **acceleration**, and **year**. What is the test error of the model obtained?

  3. (**2 pts**) Perform QDA on the training data in order to classify **mpg01** using the same variables in part 3. What is the test error of the model obtained?

  4. (**2 pts**) Perform logistic regression on the training data in order to classify **mpg01** using the same variables in part 3. What is the test error of the model obtained?

  5. (**3 pts**) Draw the ROC curves of LDA, QDA and logistic regression on the test data. Compute their AUCs and comment on which classifier you would choose. (You may find the `R` package `pROC` useful.)

- **Problem 1 (3 pts)**
  Consider the classification problem with the label of Y belong to $\mathcal{C} := \{1, 2, \ldots, K\}$ and any realization x of $X \in \mathbb{R}^p$. Let f be any classifier that maps any $x \in \mathbb{R}^p$ to a label in $\mathcal{C}$.

  1. (**2 pts**) Prove that the best function f^* (i.e. the Bayes classifier)

  $$f^* := \underset{f:\mathbb{R}^p \to \mathcal{C}}{\operatorname{argmin}} \ \mathbb{E}\Big[1\{Y \neq f(X)\} \mid X = x\Big]$$

  satisfies

  $$f^*(x) = \underset{k \in \mathcal{C}}{\operatorname{argmax}} \ \mathbb{P}(Y = k \mid X = x). \tag{0.1}$$

  2. (**1 pt**) Argue that the Bayes error equals to

  $$\mathbb{E}\Big[1\{Y \neq f^*(X)\} \mid X = x\Big] = 1 - \max_{k \in \mathcal{C}} \ \mathbb{P}(Y = k \mid X = x).$$

1. Since according to the definition of Bayes classifier, we could know that for any $X=x$, $f^*(x) = j$, if $j = \underset{k\in C}{\operatorname{argmax}} P\{Y=k|X=x\}$

and in our case we have $f^* := \underset{f:\mathbb{R}^p \to C}{\operatorname{argmin}} E[\underbrace{1\{Y \neq f(x)\}}| X=x]$,

then $f^*(x) = 1 \times \{Y \neq f(x)\} = \begin{cases} 1, & \text{if } Y \neq f(x) \\ 0, & \text{otherwise} \end{cases}$,

then $\bar{E}[1\{Y \neq f(x)\} | X=x]$ we know is

$= 1 \times P(Y \neq f(x) | X=x) + \cancel{0 \times P(Y = f(x) | X=x)}$

$= P(Y \neq f(x) | X=x)$

then f^* will become

$$f^* := \underset{f:R^p \to C}{\arg\min}\ P(Y \neq f(x) \mid X = x)$$

$$\text{then} \to = \underset{f:R^p \to C}{\arg\min} \{1 - P(Y = f(x) \mid X = x)\}$$

Since according to the definition of argmin and argmax, we know that when argmax $\{P(Y = f(x) \mid X = x)\}$ will create the equal idea of argmin $\{1 - \underbrace{P(Y = f(x) \mid X = x)}_{\text{when argmax will max it}}\}$.

$$\text{therefore } \underset{f:R^p \to C}{\arg\min} \{1 - P(Y = f(x) \mid X = x)\}$$

$$= \underset{f:R^p \to C}{\arg\max}\ P(Y = f(x) \mid X = x)$$

Since we know $C := \{1, 2, \cdots, k\}$, and $f(x) \in C$

then $\underset{k \in C}{\operatorname{argmax}}\ P(Y = f(x) \mid X = x)$

$= \underset{k \in C}{\operatorname{argmax}}\ P(Y = k \mid X = x)$

2. Since, $f^*(x) = 1 \times \{Y \neq f(x)\} = \begin{cases} 1, & \text{if } Y \neq f(x) \\ 0, & \text{otherwise} \end{cases}$,

$$E[1 \times \{Y \neq f(x)\} \mid X = x] = P(Y \neq f^*(x) \mid X = x)$$

$$= 1 - P(Y = f^*(x) \mid X = x)$$

Also, for any $X = x$, $f^*(x) = j$ if

$j = \underset{k \in C}{\operatorname{argmax}}\ P(Y = k \mid X = x)$, and therefore

the Bayes error $= E[1\{Y \neq f^*(x)\} \mid X = x]$

$$= 1 - \max_{k \in C} P(Y = k \mid X = x)$$

- **Problem 2 (3 pts)**

Consider a classification problem. Assume that the response variable Y can only take value in $\mathcal{C} = \{1, 2, 3\}$. For a fixed x_0, assume that the conditional probability of Y given $X = x_0$ follows

$$\mathbb{P}(Y = 1 \mid X = x_0) = 0.6; \quad \mathbb{P}(Y = 2 \mid X = x_0) = 0.3; \quad \mathbb{P}(Y = 3 \mid X = x_0) = 0.1.$$

Consider a naive classifier $\hat{f}$, called random guessing, which randomly picks one label from $\mathcal{C} = \{1, 2, 3\}$ with equal probability.

1. (**2 pts**) Compute the expected test error rate of $\hat{f}$ at $X = x_0$.

2. (**1 pt**) Compute the Bayes error rate at $X = x_0$ and compare it with that of $\hat{f}$.

1. Since we know a fixed X_0, and we can know that $1 \times \{Y \neq \hat{f}(X_0)\} = \begin{cases} 1, & \text{if } Y \neq \hat{f}(X_0) \\ 0, & \text{otherwise} \end{cases}$ then

the expect test error rate of $\hat{f}$ at $X = X_0$ will be

$E[1 \times \{Y \neq \hat{f}(X_0)\}] = P(Y \neq \hat{f}(X_0))$, therefore

since $1 \times \{Y \neq \hat{f}(X_0)\}$ is 1, then $P(Y \neq \hat{f}(X_0))$

$= \frac{2}{3}$

2. Since we know that the definition of Bayes error rate of f^* at $X=x$ is

$$E[1\{Y \neq f^*(x)\} \mid X=x] = 1 - \max_{1 \leq j \leq K} P\{Y=j \mid X=x\},$$

then in our situation, we could know that

$$E[1\{Y \neq f^*(X) \mid X=x_0\}]$$

$$= 1 - \max_{1 \leq j \leq 3} P(Y=j \mid X=x_0)$$

$$= 1 - P(Y=1 \mid X=x_0)$$

$$= 1 - 0.6$$

$$= 0.4 < \frac{2}{3}$$

Therefore, the Bayes error rate is smaller than expected test error rate of $\hat{f}$ at $X=x_0$

Question 3a

Since $J(\beta, \beta_0) = -\frac{1}{n}\sum_{i=1}^{n} \{ y_i \longrightarrow$

$\log\left(\frac{e^{\beta_0 + x_i^T\beta}}{1+e^{\beta_0 + x_i^T\beta}}\right) + (1-y_i)\log\left(1-\frac{e^{\beta_0 + x_i^T\beta}}{1+e^{\beta_0 + x_i^T\beta}}\right)\} + \frac{\lambda}{2}\|\beta\|_2^2$

then $\frac{\partial J(\beta,\beta_0)}{\partial \beta_0} = -\frac{1}{n}\sum_{i=1}^{n} \{ y_i \cdot \frac{1+e^{\beta_0 + x_i^T\beta}}{e^{\beta_0 + x_i^T\beta}} \cdot \longrightarrow$

$\frac{(1+e^{\beta_0 + x_i^T\beta})(e^{\beta_0 + x_i^T\beta}) - (e^{\beta_0 + x_i^T\beta})(e^{\beta_0 + x_i^T\beta})}{(1+e^{\beta_0 + x_i^T\beta})^2} + (1-y_i)(1+e^{\beta_0 + x_i^T\beta}) \cdot$

$[-(1+e^{\beta_0 + x_i^T\beta})^{-2}(e^{\beta_0 + x_i^T\beta})]\}$

$= -\frac{1}{n}\sum_{i=1}^{n} \{ y_i \cdot \frac{1}{1+e^{\beta_0 + x_i^T\beta}} - (1-y_i)\cdot \frac{e^{\beta_0 + x_i^T\beta}}{1+e^{\beta_0 + x_i^T\beta}} \}$

$= -\frac{1}{n}\sum_{i=1}^{n} \{ y_i - \frac{e^{\beta_0 + x_i^T\beta}}{1+e^{\beta_0 + x_i^T}} \}$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[-y_i + \frac{e^{\beta_0 + x_i^T\beta}}{1+e^{\beta_0+x_i^T\beta}}\right]$$

Therefore $\left.\frac{\partial J(\beta,\beta_0)}{\partial \beta_0}\right|_{\bar{\beta},\bar{\beta_0}} = \frac{1}{n}\sum_{i=1}^{n}\left[-y_i + \frac{e^{\bar{\beta_0} + x_i^T\bar{\beta}}}{1+e^{\bar{\beta_0}+x_i^T\bar{\beta}}}\right]$

b + c

```r
Evaluate <- function(true_label, pred_label) {
  #  Compute the 0-1 loss between two vectors
  #
  #  @param true_label: A vector of true labels with length n
  #  @param pred_label: A vector of predicted labels with length n
  #  @return: fraction of points get misclassified

  #########################################################################
  #  TODO                                                                 #
  #########################################################################

  error <- mean(true_label != pred_label)

  #########################################################################
  #                         END OF YOUR CODE                              #
  #########################################################################
  return(error)
}
```

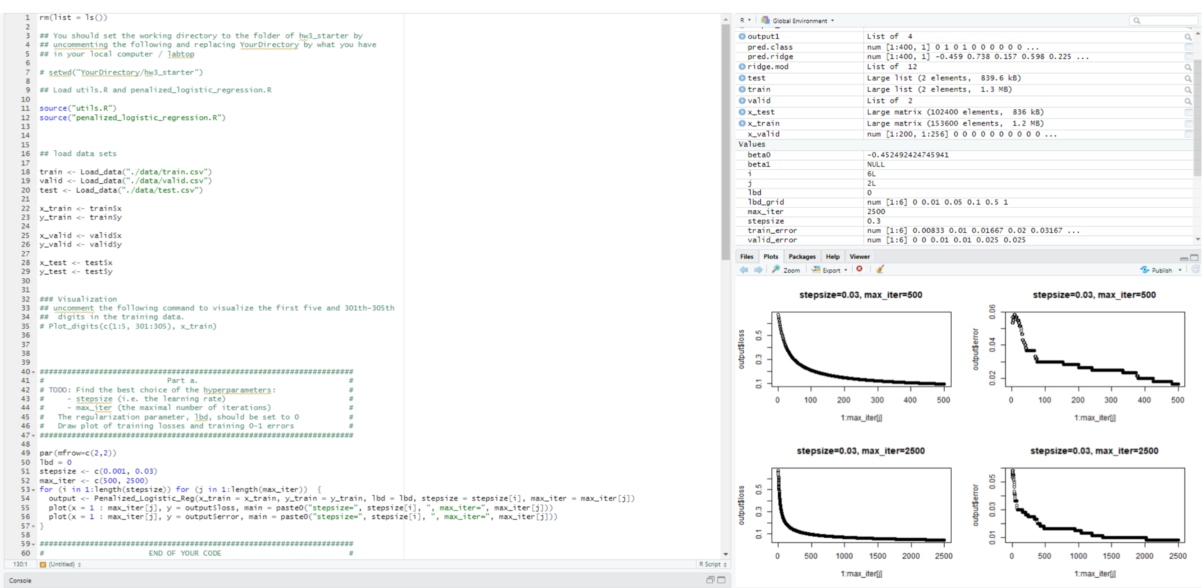
```r
Predict_logis <- function(data_feature, beta, beta0, type) {
  # Predict by the logistic classifier.
  #
  # Note: n is the number of examples
  #       p is the number of features per example
  #
  # @param data_feature: A matrix with dimension n x p, where each row corresponds to
  #   one data point.
  # @param beta: A vector of coefficients with length equal to p.
  # @param beta0: the intercept.
  # @param type: a string value within {"logit", "prob", "class"}.
  # @return: A vector with length equal to n, consisting of
  #   predicted logits,         if type = "logit";
  #   predicted probabilities,  if type = "prob";
  #   predicted labels,         if type = "class".

  n <- nrow(data_feature)
  pred_vec <- rep(0, n)

  #########################################################################
  #  TODO                                                                 #
  #########################################################################
  if (type == "logit") {
    pred_vec <- beta0 + data_feature %*% beta
  } else if (type == "prob") {
    pred_vec <- exp(beta0 + data_feature %*% beta) / (1 + exp(beta0 + data_feature %*% beta))
  } else if (type == "class") {
    pred_vec <- ifelse(exp(beta0 + data_feature %*% beta) / (1 + exp(beta0 + data_feature %*% beta)) > 0.5, 1, 0)
  }
  pred_vec <- as.numeric(pred_vec)

  #########################################################################
  #                         END OF YOUR CODE                              #
  #########################################################################

  return(pred_vec)
}
```

```r
Comp_gradient <- function(data_feature, data_label, beta, beta0, lbd) {
  # Compute and return the gradient of the penalized logistic regression
  #
  # Note: n is the number of examples
  #       p is the number of features per example
  #
  # @param data_feature: A matrix with dimension n x p, where each row corresponds to
  #   one data point.
  # @param data_label: A vector of labels with length equal to n.
  # @param beta: A vector of coefficients with length equal to p.
  # @param beta0: the intercept.
  # @param lbd: the regularization parameter
  #
  # @return: a (p+1) x 1 vector of gradients, the first coordinate is the gradient
  #   w.r.t. the intercept.

  n <- nrow(data_feature)
  p <- ncol(data_feature)
  grad <- rep(0, 1 + p)

  #######################################################################
  # TODO:                                                               #
  #######################################################################
  grad[1] <- mean(-data_label + exp(beta0 + data_feature %*% beta) / (1 + exp(beta0 + data_feature %*% beta)))
  grad[2:(1+p)] <- rowMeans(t(data_feature) %*% diag(as.numeric(-data_label + exp(beta0 + data_feature %*% beta)/(1+exp(beta0 + data_feature %*% beta))))) + lbd * beta

  #######################################################################
  #                        END OF YOUR CODE                             #
  #######################################################################
  return(grad)
}
```

```r
Comp_loss <- function(data_feature, data_label, beta, beta0, lbd) {
  # Compute and return the loss of the penalized logistic regression
  #
  # Note: n is the number of examples
  #       p is the number of features per example
  #
  # @param data_feature: A matrix with dimension n x p, where each row corresponds to
  #   one data point.
  # @param data_label: A vector of labels with with length equal to n.
  # @param beta: A vector of coefficients with length equal to p.
  # @param beta0: the intercept.
  # @param lbd: the regularization parameter
  #
  # @return: a value of the loss function

  #######################################################################
  # TODO:                                                               #
  #######################################################################

  p <- exp(beta0 + data_feature %*% beta) / (1 + exp(beta0 + data_feature %*% beta))
  loss <- -mean(data_label * log(p) + (1 - data_label)*log(1-p)) + (lbd/2) * sqrt(sum(beta^2))

  #######################################################################
  #                        END OF YOUR CODE                             #
  #######################################################################
  return(loss)
}
```

```r
Penalized_Logistic_Reg <- function(x_train, y_train, lbd, stepsize, max_iter) {
  # This is the main function to fit the Penalized Logistic Regression
  #
  # Note: n is the number of examples
  #       p is the number of features per example
  #
  # @param x_train: A matrix with dimension n x p, where each row corresponds to
  #   one training point.
  # @param y_train: A vector of labels with length equal to n.
  # @param lbd: the regularization parameter.
  # @param stepsize: the learning rate.
  # @param max_iter: a positive integer specifying the maximal number of
  #   iterations.
  #
  # @return: a list containing four components:
  #   loss: a vector of loss values at each iteration
  #   error: a vector of 0-1 errors at each iteration
  #   beta: the estimated p coefficient vectors
  #   beta0: the estimated intercept.

  p <- ncol(x_train)

  # Initialize parameters to 0
  beta_cur <- rep(0, p)
  beta0_cur <- 0

  # Create the vectors for recording values of loss and 0-1 error during
  # the training procedure
  loss_vec <- rep(0, max_iter)
  error_vec <- rep(0, max_iter)

  #######################################################################
  # TODO:                                                               #
  # Modify this section to perform gradient descent and to compute     #
  # losses and 0-1 errors at each iterations.                           #
  #######################################################################

  for (i in 1:max_iter) {
    gra1 <- Comp_gradient(data_feature = x_train, data_label = y_train, beta = beta_cur, beta0 = beta0_cur, lbd = lbd)
    beta0_cur <- beta0_cur - stepsize * gra1[1]
    beta_cur <- beta_cur - stepsize * gra1[-1]
    loss_vec[i] <- Comp_loss(data_feature = x_train, data_label = y_train, beta = beta_cur, beta0 = beta0_cur, lbd = lbd)
    pred_class <- Predict_logis(data_feature = x_train, beta = beta_cur, beta0 = beta0_cur, type = "class")
    error_vec[i] <- Evaluate(true_label = y_train, pred_label = pred_class)
  }

  #######################################################################
  #                          END OF YOUR CODE                           #
  #######################################################################

  return(list("loss" = loss_vec, "error" = error_vec,
              "beta" = beta_cur, "beta0" = beta0_cur))
}
```

4.



In this question, I choose the max iter to be 2500 and stepsize to be 0.03, and find it converge to 0 for a long distance and means it is well-fitted and is approcate for the case. Also, training 0-1 error has the same pattern with training loss.

5.

```
Evaluate <- function(true_label, pred_label) {
  #  Compute the 0-1 loss between two vectors
  #
  #  @param true_label: A vector of true labels with length n
  #  @param pred_label: A vector of predicted labels with length n
  #  @return: fraction of points get misclassified

  #######################################################################
  #  TODO                                                               #
  #######################################################################

  error <- mean(true_label != pred_label)

  #######################################################################
  #                          END OF YOUR CODE                           #
  #######################################################################
  return(error)
}
```

In this question, I choose the max iter to be 2500 and stepsize to be 0.03, find that the train error and valid error relationship appears as cubic and approach 0 as grid approach 0. The best value of λ is 0.

6.

```r
########################################################################
#                              Part c.                                 #
# TODO: using the best stepsize,  max_iter and lbd you found, fit     #
#  the penalized logistic regression and compute its test 0-1 error #
########################################################################


stepsize <- 0.3  # this should be replaced by your answer in Part a
max_iter <- 2500  # this should be replaced by your answer in Part a
lbd <- 0       # this should be replaced by your answer in Part b

output1 <- Penalized_Logistic_Reg(x_train = x_train, y_train = y_train, lbd = lbd, stepsize = stepsize, max_iter = max_iter)
beta1 <- output1$beta1
beta0 <- output1$beta0
pred_class <- Predict_logis(data_feature = x_test, beta1 = beta1, beta0 = beta0, type = "class")
error <- Evaluate(true_label = y_test, pred_label = pred_class)
error
library(glmnet)
ridge.mod <- glmnet(x_train, y_train, alpha = 0, lambda = lbd/2)
pred.ridge <- predict(ridge.mod, s = lbd, newx = x_test)
pred.class <- ifelse(pred.ridge > 0.5, 1, 0)
mean(pred.class != y_test)

########################################################################
#                        END OF YOUR CODE                              #
########################################################################
```

In this question, one error is 0.0225, another is 0.0525, then we know glmnet is worse than the previous one because of higher error rate.

STA314 hw3 problem 4

Problem 4

1

```
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
data(Auto)
Auto <- Auto %>% mutate(mpg01 = ifelse(mpg > median(mpg), 1, 0))
num_samples <- nrow(Auto)
set.seed(0)
train_index <- sample(num_samples,num_samples*0.7)
train <- Auto[train_index, ]
test <- Auto[-train_index, ]
```

2

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
lda.fit <- lda(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year, data = tra
lda.class <- predict(lda.fit, test)$class
mean(lda.class != test$mpg01)
```

```
## [1] 0.1186441
```

The test error here is 0.1186441.

3

```r
qda.fit <- qda(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year, data = tra
qda.class <- predict(qda.fit, test)$class
mean(qda.class != test$mpg01)
```

```
## [1] 0.1101695
```

The test error here is 0.1101695.

4

```r
glm.fit <- glm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year, data = tra
glm.probs <- predict(glm.fit, test, type="response")
glm.pred <- ifelse(glm.probs > 0.5, 1, 0)
mean(glm.pred != test$mpg01)
```

```
## [1] 0.1016949
```

The test error here is 0.1016949.

5

LDA:

```r
library(pROC)
```

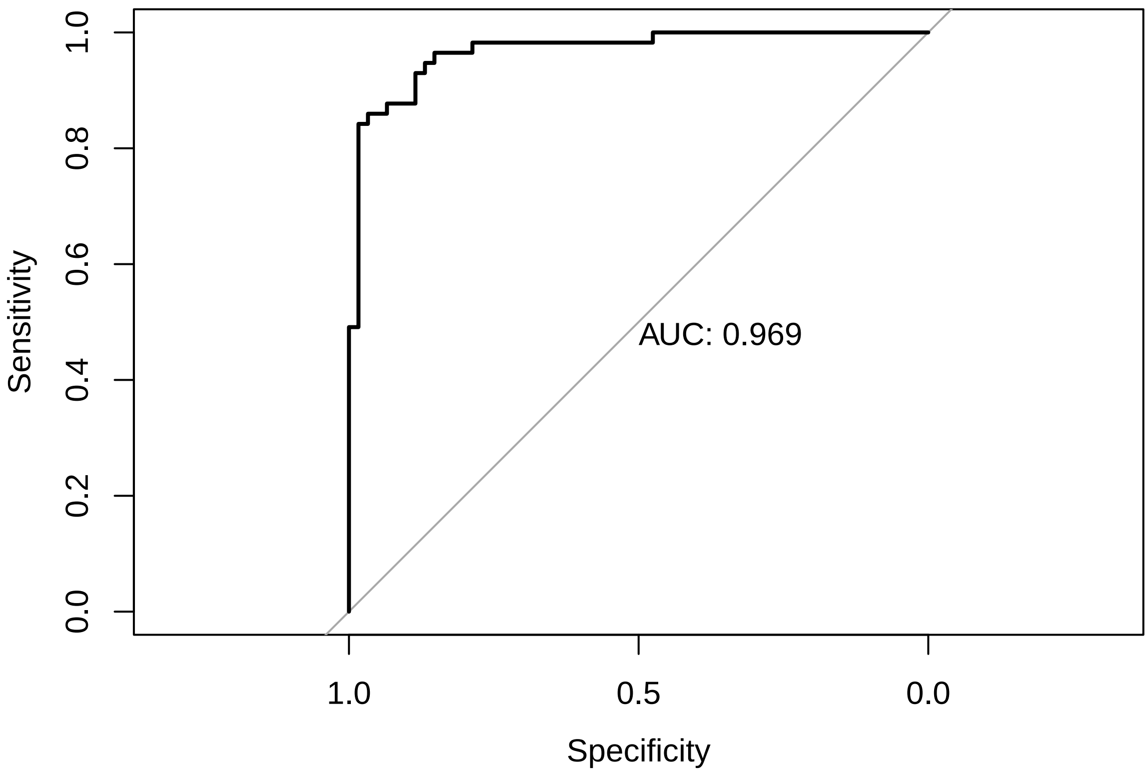
```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
lda.pred <- predict(lda.fit, test)
roc(test$mpg01~lda.pred$posterior[,2],plot=TRUE,print.auc=TRUE)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

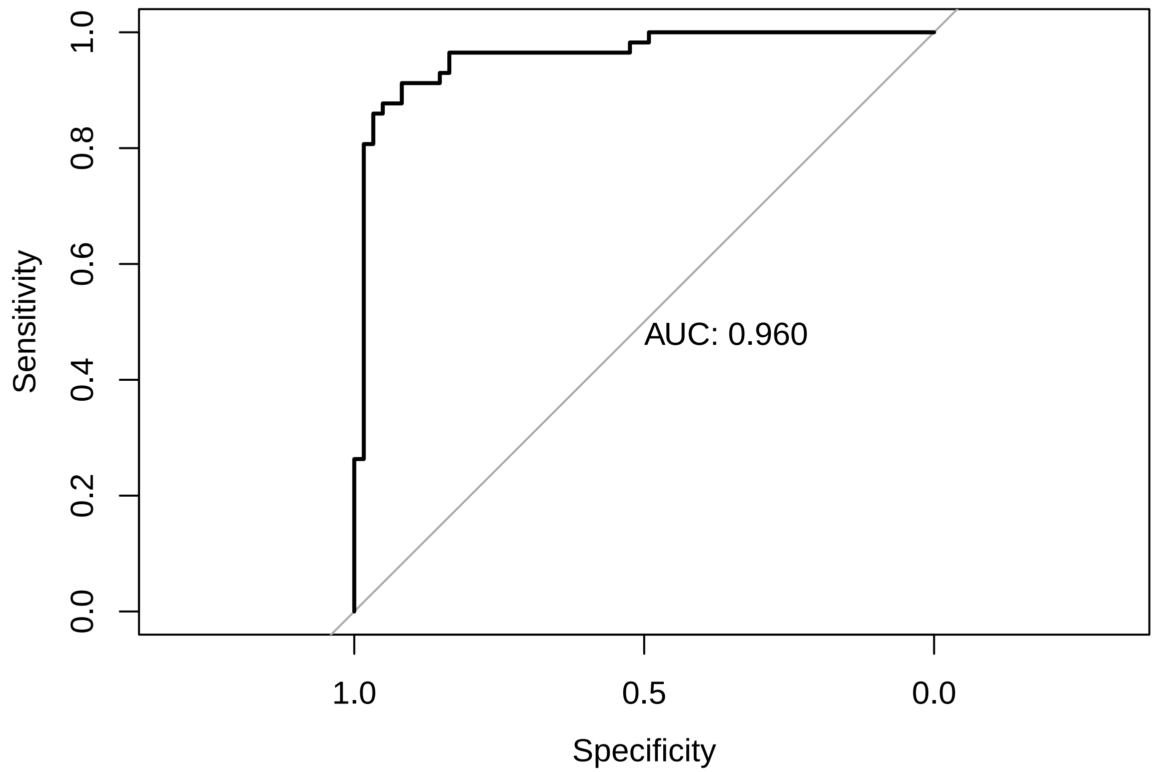



```
##
## Call:
## roc.formula(formula = test$mpg01 ~ lda.pred$posterior[, 2], plot = TRUE,     print.auc = TRUE)
##
## Data: lda.pred$posterior[, 2] in 61 controls (test$mpg01 0) < 57 cases (test$mpg01 1).
## Area under the curve: 0.9687
```

QDA:

```r
qda.pred = predict(qda.fit, test)
roc(test$mpg01~qda.pred$posterior[,2],plot=TRUE,print.auc=TRUE)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



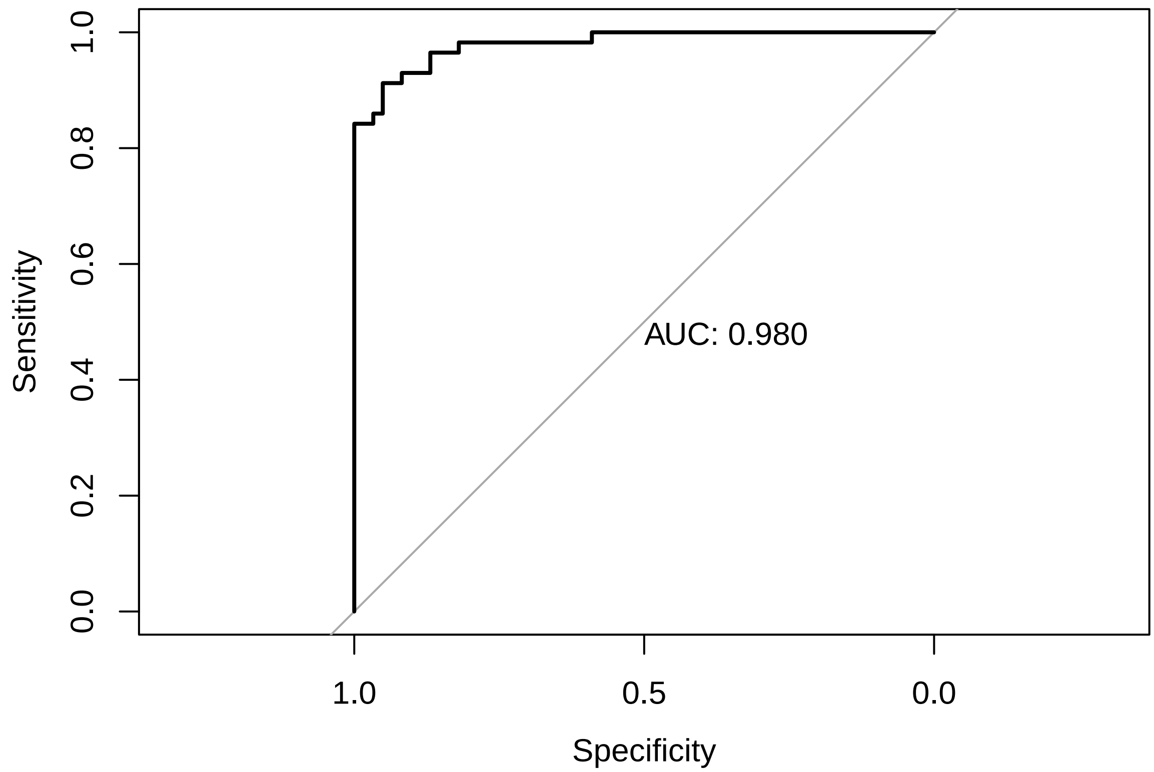

```
##
## Call:
## roc.formula(formula = test$mpg01 ~ qda.pred$posterior[, 2], plot = TRUE,     print.auc = TRUE)
##
## Data: qda.pred$posterior[, 2] in 61 controls (test$mpg01 0) < 57 cases (test$mpg01 1).
## Area under the curve: 0.96
```

Logistic regression:

```
roc(test$mpg01~glm.probs,plot=TRUE,print.auc=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.formula(formula = test$mpg01 ~ glm.probs, plot = TRUE, print.auc = TRUE)
##
## Data: glm.probs in 61 controls (test$mpg01 0) < 57 cases (test$mpg01 1).
## Area under the curve: 0.9804
```

In this case, I will choose the logistic regression classifier since it has the highest AUC.