Breast Cancer Diagnosis

Hongjie Liu, Xicheng Xie, Jiajun Tao, Shaohan Chen, Yujia Li

April 3rd, 2023

Outline

- Background
- Task Introduction
- Task 1
- Task 2
- Task 3
- Task 4
- Discussions
- Reference
- Q&A

Background

Breast Cancer Diagnosis:

- In this project we study the breast cancer diagnosis problem
- The goal of the exercise is to build a predictive model based on logistic regression to facilitate cancer diagnosis
- We move towards the goal with the steps of task 1, 2, 3 and 4.

Background

Data Source:

- The data is the breast cancer medical data retrieved from "breast-cancer.csv", which has 569 rows and 32 columns
- The first column `ID` labels individual breast tissue images; The second column `Diagnosis` identifies if the image is coming from cancer tissue or benign cases (M = malignant, B = benign). There are 357 benign and 212 malignant cases
- The other 30 columns correspond to mean, standard deviation and the largest values (points on the tails) of the distributions of 10 features computed for the cellnuclei

Task Introduction

- Task 1: Build a logistic model to classify the images into malignant/benign, and write down your likelihood function, its gradient and Hessian matrix.
- Task 2: Develop a Newton-Raphson algorithm to estimate your model.
- Task 3: Build a logistic-LASSO model to select features, and implement a path-wise coordinate-wise optimization algorithm to obtain a path of solutions with a sequence of descending λ's.
- Task 4: Use 5-fold cross-validation to select the best λ. Compare the prediction performance between the 'optimal' model and 'full' model.

Task 1 - Objective

Objective:

Build a logistic model to classify the images into malignant/benign, and write down your likelihood function, its gradient and Hessian matrix.

Task 1 - Build a logistic model

"Diagnosis" variable will be coded as 1 for malignant cases and 0 for benign cases.

Given n i.i.d. observations with p predictors, we consider a logistic regression model

$$P(Y_i = 1 \mid \mathbf{x}_i) = \frac{e^{\mathbf{x}_i^\top \beta}}{1 + e^{\mathbf{x}_i^\top \beta}}, \ i = 1, \ldots, n \tag{1}$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^\top \in \mathbb{R}^{p+1}$ is the parameter vector, $\mathbf{x}_i = (1, X_{i1}, \ldots, X_{ip})^\top$ is the vector of predictors in the i-th observation, and $Y_i \in \{0, 1\}$ is the binary response in the i-th observation.

Task 1 - Build a logistic model

Let $\mathbf{y} = (Y_1, Y_2, \ldots, Y_n)^\top$ denote the response vector, and $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times (p+1)}$ denote the design matrix.

The observed likelihood of $\{(Y_1, \mathbf{x}_1), (Y_2, \mathbf{x}_2) \ldots, (Y_n, \mathbf{x}_n)\}$ is

$$L(\beta; \mathbf{y}, \mathbf{X}) = \prod_{i=1}^{n} \left[\left(\frac{e^{\mathbf{x}_i^\top \beta}}{1 + e^{\mathbf{x}_i^\top \beta}} \right)^{Y_i} \left(\frac{1}{1 + e^{\mathbf{x}_i^\top \beta}} \right)^{1 - Y_i} \right]$$

Task 1 - Build a logistic model

Maximizing the likelihood is equivalent to maximizing the log-likelihood function:

$$f(\beta; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \left[Y_i \mathbf{x}_i^\top \beta - \log \left(1 + e^{\mathbf{x}_i^\top \beta} \right) \right]. \tag{2}$$

The estimates of model parameters are

$$\widehat{\beta} = \arg\max_{\beta} \ f(\beta; \mathbf{y}, \mathbf{X}),$$

and the optimization problem is

$$\max_{\beta} \ f(\beta; \mathbf{y}, \mathbf{X}). \tag{3}$$

Task 1 - Build a logistic model

Denote $p_i = P(Y_i = 1 \mid \mathbf{x}_i)$ as given in (1) and $\mathbf{p} = (p_1, p_2, \ldots, p_n)^\top$. The gradient of f is:

$$\begin{aligned}
\nabla f(\beta; \mathbf{y}, \mathbf{X}) &= \mathbf{X}^\top (\mathbf{y} - \mathbf{p}) \\
&= \sum_{i=1}^{n} (Y_i - p_i)\mathbf{x}_i \\
&= \begin{pmatrix} \sum_{i=1}^{n} (Y_i - p_i) \\ \sum_{i=1}^{n} (Y_i - p_i) X_{i1} \\ \vdots \\ \sum_{i=1}^{n} (Y_i - p_i) X_{ip} \end{pmatrix}
\end{aligned}$$

Task 1 - Build a logistic model

Denote $w_i = p_i(1 - p_i) \in (0, 1)$ and $\mathbf{W} = \mathrm{diag}(w_1, \ldots, w_n)$. The Hessian matrix of f is given by

$$\begin{aligned}
\nabla^2 f(\beta; \mathbf{y}, \mathbf{X}) &= -\mathbf{X}^\top \mathbf{W} \mathbf{X} \\
&= -\sum_{i=1}^{n} w_i \mathbf{x}_i \mathbf{x}_i^\top \\
&= -\begin{pmatrix}
\sum_{i=1}^n w_i & \sum_{i=1}^n w_i X_{i1} & \cdots & \sum_{i=1}^n w_i X_{i1} \\
\sum_{i=1}^n w_i X_{i1} & \sum_{i=1}^n w_i X_{i1}^2 & \cdots & \sum_{i=1}^n w_i X_{i1} X_{ip} \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{i=1}^n w_i X_{ip} & \sum_{i=1}^n w_i X_{ip} X_{i1} & \cdots & \sum_{i=1}^n w_i X_{ip}^2
\end{pmatrix}
\end{aligned}$$

Task 1 - Build a logistic model

Next, we show that the Hessian matrix $\nabla^2 f(\beta; \mathbf{y}, \mathbf{X})$ is a negative-definite matrix if $\mathbf{X}$ has full rank.

Proof. For any $(p+1)$-dimensional nonzero vector $\boldsymbol{\alpha}$, given that $\mathbf{X}$ has full rank, $\mathbf{X}\boldsymbol{\alpha}$ is also a nonzero vector. Since $\mathbf{W}$ is positive-definite, we have

$$\begin{aligned}\boldsymbol{\alpha}^\top \nabla^2 f(\beta; \mathbf{y}, \mathbf{X})\boldsymbol{\alpha} &= \boldsymbol{\alpha}^\top(-\mathbf{X}^\top \mathbf{W}\mathbf{X})\boldsymbol{\alpha} \\ &= -(\mathbf{X}\boldsymbol{\alpha})^\top \mathbf{W}(\mathbf{X}\boldsymbol{\alpha}) \\ &< 0.\end{aligned}$$

Thus, $\nabla^2 f(\beta; \mathbf{y}, \mathbf{X})$ is negative-definite. □

Hence, the optimization problem (3) is a well-defined problem.

Task 2 - Objective

Objective:

Develop a Newton-Raphson algorithm to estimate your model

- Recall The target function f given in task 1 is:

$$f(\beta; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \left[Y_i \mathbf{x}_i^\top \beta - \log\left(1 + e^{\mathbf{x}_i^\top \beta}\right) \right].$$

- We develop a modified Newton-Raphson algorithm including a step-halving step to maximize the target function.

Task 2 - Newton-Raphson algorithm design

Algorithm 1 Newton-Raphson algorithm

Require: $f(\beta)$ - target function as given in (2); β_0 - starting value
Ensure: $\widehat{\beta}$ such that $\widehat{\beta} \approx \arg\max_\beta \ f(\beta)$

$i \leftarrow 0$, where i is the current number of iterations
$f(\beta_{-1}) \leftarrow -\infty$
while convergence criterion is not met **do**
    $i \leftarrow i + 1$
    $\mathbf{d}_i \leftarrow -[\nabla^2 f(\beta_{i-1})]^{-1} \nabla f(\beta_{i-1})$, where $\mathbf{d}_i$ is the direction in the i-th iteration
    $\lambda_i \leftarrow 1$, where λ_i is the multiplier in the i-th iteration
    $\beta_i \leftarrow \beta_{i-1} + \lambda_i \mathbf{d}_i$
    while $f(\beta_i) \leq f(\beta_{i-1})$ **do**
        $\lambda_i \leftarrow \lambda_i / 2$
        $\beta_i \leftarrow \beta_{i-1} + \lambda_i \mathbf{d}_i$
    end while
end while
$\widehat{\beta} \leftarrow \beta_i$

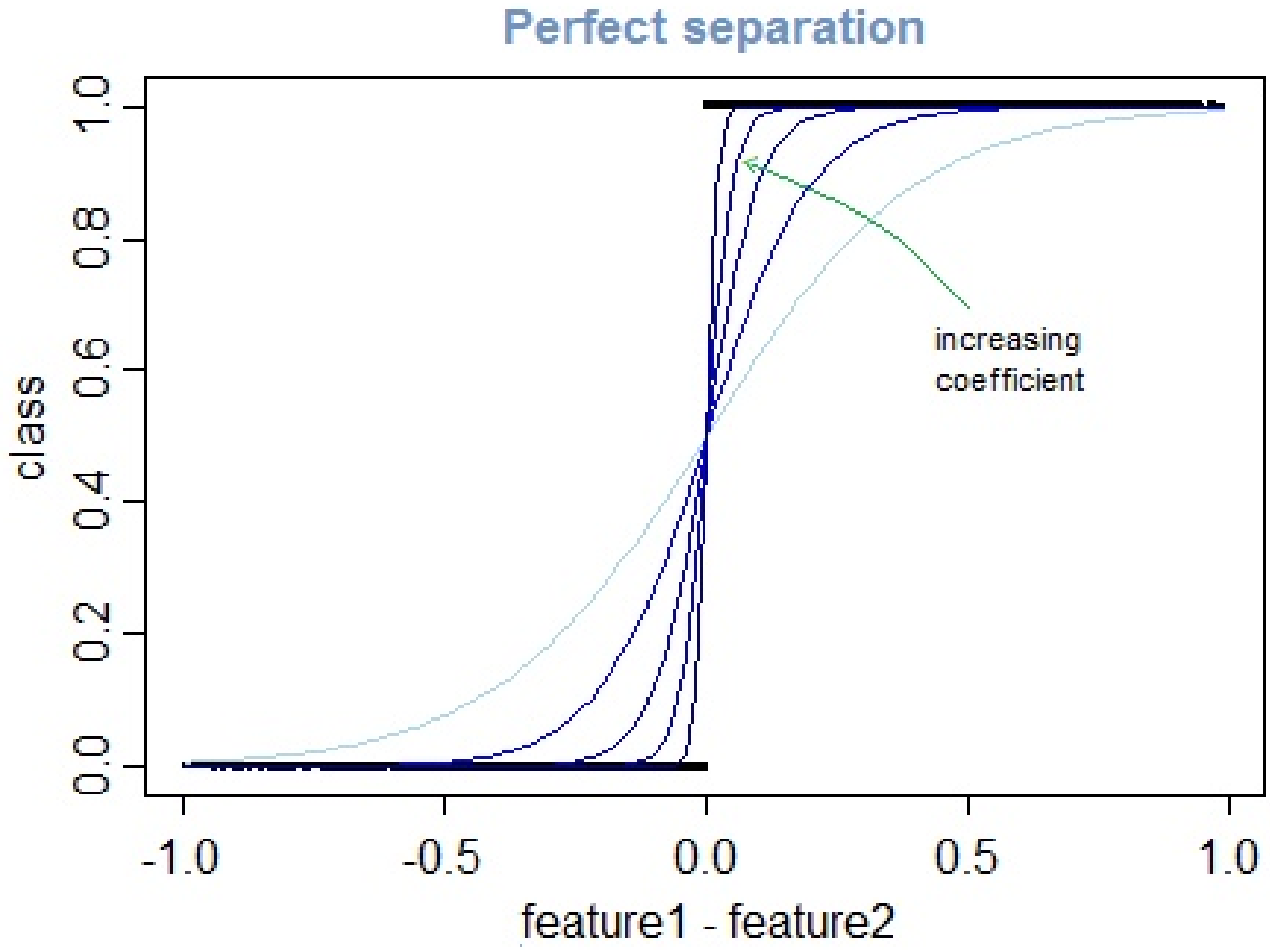
Task 2 - Newton-Raphson algorithm R code

```r
NewtonRaphson <- function(dat, func, start, tol = 1e-8, maxiter = 200) {
  i <- 0
  cur <- start
  stuff <- func(dat, cur)
  res <- c(0, stuff$f, cur)
  prevf <- -Inf
  X <- cbind(rep(1, nrow(dat)), as.matrix(dat[, -1]))
  y <- dat[, 1]
  warned <- 0
  while (abs(stuff$f - prevf) > tol && i < maxiter) {
    i <- i + 1
    prevf <- stuff$f
    prev <- cur
    d <- -solve(stuff$Hess) %*% stuff$grad
    cur <- prev + d
    lambda <- 1
    maxhalv <- 0
    while (func(dat, cur)$f < prevf && maxhalv < 50) {
      maxhalv <- maxhalv + 1
      lambda <- lambda / 2
      cur <- prev + lambda * d
    }
    stuff <- func(dat, cur)
    res <- rbind(res, c(i, stuff$f, cur))
    y_hat <- ifelse(X %*% cur > 0, 1, 0)
    if (warned == 0 && sum(y - y_hat) == 0) {
      warning("Complete separation occurs. Algorithm does not converge.")
      warned <- 1
    }
  }
  colnames(res) <- c("iter", "target_function", "(Intercept)", names(dat)[-1])
  return(res)
}
```

Task 2 - Complete separtion

- Sometimes our algorithm does not converge because of the complete separation.
- A complete separation in a logistic regression, sometimes also referred to as perfect prediction, occurs whenever there exists some vector of coefficients β such that $Y_i = 1$ whenever $\mathbf{x}_i^\top \beta > 0$ and $Y_i = 0$ whenever $\mathbf{x}_i^\top \beta \leq 0$.
- Complete separation occurs when a linear function of predictors can perfectly classify the response.

Task 2 - Complete separation



Task 2 - Complete separation

- We have proved that: when there exists a vector of coefficients $\hat{\beta}$ such that $Y_i = 1$ whenever $\mathbf{x}_i^\top \hat{\beta} > 0$ and $Y_i = 0$ whenever $\mathbf{x}_i^\top \hat{\beta} \leq 0$, there does not exist $\beta^* \in \mathbb{R}^{(p+1)}$ such that $\beta^* = \arg\max_\beta f(\beta)$, where f is given in (2). Thus our algorithm does not converge. (proof is attached in report appendix)
- If there is no complete separation, the parameters output by `glm` function and our algorithm are demonstrated to be the same.
- In this case, given the perfect separation does exist, we choose the β based on the pre-defined `tol = 1e-8` in the algorithm.
- The difference of β in our algorithm output and `glm` function is out of the different termination condition.

Task 2 - Comparison

Comparison of using `glm` function and our algorithm (part of)

predictor	ours	glm
(Intercept)	111.7230206	90.9690365
radius_mean	-3646.8235387	-2560.3938902
texture_mean	-2.8949199	0.8812037
perimeter_mean	1257.9481173	789.2724398
area_mean	2091.5001210	1539.8345650
smoothness_mean	180.4591375	128.8762247
compactness_mean	-471.3749334	-346.6691873
concavity_mean	-0.3063034	9.0810082
concave.points_mean	287.3555092	215.4808031
symmetry_mean	10.8459784	10.4772590
fractal_dimension_mean	-40.6413497	-28.6916549
radius_se	854.4290933	617.5687358
texture_se	-105.8920782	-79.9788638
perimeter_se	-1231.6514585	-917.3916527
area_se	789.0550146	628.6222313
smoothness_se	-83.3449730	-63.7660367
compactness_se	473.1970464	344.3180183
concavity_se	-440.6629325	-323.8533730
concave.points_se	503.3728692	374.7610875
symmetry_se	-144.4580558	-108.6211792

Task 3 - Objective

Objective:

Build a logistic-LASSO model to select features by implementing a path-wise coordinate-wise optimization algorithm.

- Log-likelihood f in task 1:

$$f(\beta; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \left[Y_i \mathbf{x}_i^\top \beta - \log \left(1 + e^{\mathbf{x}_i^\top \beta} \right) \right].$$

- LASSO estimates the logistic model parameters β by optimizing a penalized loss function:

$$\min_{\beta} \; -\frac{1}{n} f(\beta) + \lambda \sum_{k=1}^{p} |\beta_k|. \tag{4}$$

where $\lambda \geq 0$ is the tuning parameter. Note that the intercept is not penalized and all predictors are standardized.

Task 3 - Outer Loop

- Compute the solutions of the optimization problem (4) for a decreasing sequence of values for λ: $\{\lambda_1, \ldots, \lambda_m\}$, starting at the smallest value $\lambda_1 = \lambda_{max}$ such that $\hat{\beta}_j = 0,\ j = 1, \ldots, p$.

$$\lambda_{max} = \max_{j \in \{1,\ldots,p\}} \left| \frac{1}{n} \sum_{i=1}^{n} X_{ij}(Y_i - \bar{Y}) \right|, \tag{5}$$

  where $\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$.
- For tuning parameter value λ_{k+1}, we initialize coordinate descent algorithm at the computed solution for λ_k (warm start).

Task 3 - Middle Loop

- For a fixed λ, find the estimates of β by solving the optimization problem (4).
- Based on current parameter estimates $\tilde{\beta}$, we form a quadratic approximation to the log-likelihood f using a Taylor expansion:

$$
\begin{aligned}
f(\beta) \approx \ell(\beta) &= f(\tilde{\beta}) + (\beta - \tilde{\beta})^\top \nabla f(\tilde{\beta}) + \frac{1}{2}(\beta - \tilde{\beta})^\top \nabla^2 f(\tilde{\beta})(\beta - \tilde{\beta}) \\
&= f(\tilde{\beta}) + [\mathbf{X}(\beta - \tilde{\beta})]^\top (\mathbf{y} - \tilde{\mathbf{p}}) - \frac{1}{2}[\mathbf{X}(\beta - \tilde{\beta})]^\top \tilde{\mathbf{W}}\mathbf{X}(\beta - \tilde{\beta}) \\
&= -\frac{1}{2}\sum_{i=1}^{n} \tilde{w}_i \left[\mathbf{x}_i^\top (\tilde{\beta} - \beta) + \frac{Y_i - \tilde{p}_i}{\tilde{w}_i}\right] + \frac{1}{2}\sum_{i=1}^{n} \tilde{w}_i \left(\frac{Y_i - \tilde{p}_i}{\tilde{w}_i}\right)^2 + f(\tilde{\beta})
\end{aligned}
$$

where $\tilde{\mathbf{p}} = (\tilde{p}_1, \ldots, \tilde{p}_n)^\top$ and $\tilde{\mathbf{W}} = \mathrm{diag}(\tilde{w}_1, \ldots, \tilde{w}_n)$ are the estimates of $\mathbf{p}$ and $\mathbf{W}$ based on $\tilde{\beta}$.

Task 3 - Middle Loop

Let $\tilde{z}_i = \mathbf{x}_i^\top \tilde{\beta} + \frac{Y_i - \tilde{p}_i}{\tilde{w}_i}$, we have

$$\ell(\beta) = -\frac{1}{2} \sum_{i=1}^{n} \tilde{w}_i (\tilde{z}_i - \mathbf{x}_i^\top \beta)^2 + C(\tilde{\beta}), \tag{6}$$

where is $\tilde{z}_i$ the working response, $\tilde{w}_i$ is the working weight, and C is a function that does not depend on β.

Task 3 - Inner Loop

With fixed $\tilde{w}_i$'s, $\tilde{z}_i$'s, and a fixed form of ℓ based on the estimates of β in the previous iteration of the middle loop, we use coordinate descent method to update β by solving

$$\min_{\beta} \; -\frac{1}{n}\ell(\beta) + \lambda \sum_{k=1}^{p} |\beta_k|,$$

which is equivalent to

$$\min_{\beta} \; \frac{1}{2n} \sum_{i=1}^{n} \tilde{w}_i (\tilde{z}_i - \mathbf{x}_i^\top \beta)^2 + \lambda \sum_{k=1}^{p} |\beta_k|, \tag{7}$$

Task 3 - Inner Loop

- Based on the current estimates $\tilde{\beta}_k$ for $k \neq j$:

$$\min_{\beta_j} \frac{1}{2n} \sum_{i=1}^{n} \tilde{w}_i \left(\tilde{z}_i - x_{ij}\beta_j - \sum_{k \neq j} x_{ik}\tilde{\beta}_k \right)^2 + \lambda|\beta_j| + \lambda \sum_{k \neq j} |\tilde{\beta}_k|.$$

- Updates:

$$\tilde{\beta}_0 \leftarrow \frac{\sum_{i=1}^{n} \tilde{w}_i(\tilde{z}_i - \sum_{k=1}^{p} x_{ik}\tilde{\beta}_k)}{\sum_{i=1}^{n} \tilde{w}_i},$$

$$\tilde{\beta}_j \leftarrow \frac{S\left(\frac{1}{n}\sum_{i=1}^{n} \tilde{w}_i x_{ij}(\tilde{z}_i - \sum_{k \neq j} x_{ik}\tilde{\beta}_k), \lambda\right)}{\frac{1}{n}\sum_{i=1}^{n} \tilde{w}_i x_{ij}^2}, \; j = 1, \ldots, p$$

  where $S(z, \gamma)$ is the soft-thresholding operator with value

$$S(z, \gamma) = \text{sign}(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma, & \text{if } z > 0 \text{ and } \gamma < |z| \\ z + \gamma, & \text{if } z < 0 \text{ and } \gamma < |z| \\ 0, & \text{if } \gamma \geq |z| \end{cases}$$

- Keep updating estimates of β_j's repeatedly for $j = 0, 1, 2, ..., p, 0, 1, 2, ...$ until convergence.

Task 3 - Algorithm Structure

Algorithm Structure:

- Outer Loop: Decrement λ.
- Middle Loop: Update the quadratic approximation ℓ using the current parameters estimates.
- Inner Loop: Run the coordinate descent algorithm on the penalized weighted-least-squares problem.

Task 3 - Algorithm Structure

Algorithm 1 Path-wise coordinate-wise optimization algorithm

Require: $g(\boldsymbol{\beta}, \lambda) = -\frac{1}{n} f(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|$ - target function, where $f(\boldsymbol{\beta})$ is given in (1); $\boldsymbol{\beta}_0$ - starting value; $\{\lambda_1, \ldots, \lambda_m\}$ - a sequence of descending λ's, where $\lambda_1 = \lambda_{max}$ is given in (3); ϵ - tolerance; N_s, N_t - maximum number of iterations of the middle and inner loops

Ensure: $\widehat{\boldsymbol{\beta}}(\lambda_r)$ such that $\widehat{\boldsymbol{\beta}}(\lambda_r) \approx \arg\min_{\boldsymbol{\beta}} \; g(\boldsymbol{\beta}, \lambda_r),\ r = 1, \ldots, m$

$\tilde{\boldsymbol{\beta}}_0(\lambda_1) \leftarrow \boldsymbol{\beta}_0$
OUTER LOOP
for $r \in \{1, \ldots, m\}$, where r is the current number of iterations of the outer loop, **do**
    $s \leftarrow 0$, where s is the current number of iterations of the middle loop
    $g(\tilde{\boldsymbol{\beta}}_{-1}(\lambda_r), \lambda_r) \leftarrow \infty$
    MIDDLE LOOP
    while $t \geq 2$ and $s < N_s$ **do**
        $s \leftarrow s + 1$
        Update $\tilde{w}_i^{(s)}$, $\tilde{z}_i^{(s)}$ $(i = 1, \ldots, n)$, and thus $\ell_s(\boldsymbol{\beta})$ as given in (4) based on $\tilde{\boldsymbol{\beta}}_{s-1}(\lambda_r)$
        $t \leftarrow 0$, where t is the current number of iterations of the inner loop
        $\tilde{\boldsymbol{\beta}}_s^{(0)}(\lambda_r) \leftarrow \tilde{\boldsymbol{\beta}}_{s-1}(\lambda_r)$
        $h_s(\tilde{\boldsymbol{\beta}}_s^{(-1)}(\lambda_r), \lambda_r) \leftarrow \infty$, where $h_s(\boldsymbol{\beta}, \lambda) = -\frac{1}{n}\ell_s(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|$
        INNER LOOP
        while $\left| h_s(\tilde{\boldsymbol{\beta}}_s^{(t)}(\lambda_r), \lambda_r) - h_s(\tilde{\boldsymbol{\beta}}_s^{(t-1)}(\lambda_r), \lambda_r) \right| > \epsilon$ and $t < N_t$ **do**
            $t \leftarrow t + 1$
            $\tilde{\beta}_0^{(t)}(\lambda_r) \leftarrow \sum_{i=1}^{n} \tilde{w}_i^{(s)} \left(\tilde{z}_i^{(s)} - \sum_{k=1}^{p} x_{ik} \tilde{\beta}_k^{(t-1)}(\lambda_r) \right) \Big/ \sum_{i=1}^{n} \tilde{w}_i^{(s)}$
            for $j \in \{1, \ldots, p\}$ **do**
                $\tilde{\beta}_j^{(t)}(\lambda_r) \leftarrow S\left(\frac{1}{n} \sum_{i=1}^{n} \tilde{w}_i^{(s)} x_{ij} \left(\tilde{z}_i^{(s)} - \sum_{k<j} x_{ik} \tilde{\beta}_k^{(t)}(\lambda_r) - \sum_{k>j} x_{ik} \tilde{\beta}_k^{(t-1)}(\lambda_r) \right), \lambda_r \right) \Big/ \frac{1}{n} \sum_{i=1}^{n} \tilde{w}_i^{(s)} x_{ij}^2$
            end for
        end while
        $\tilde{\boldsymbol{\beta}}_s(\lambda_r) \leftarrow \tilde{\boldsymbol{\beta}}_s^{(t)}(\lambda_r)$
    end while
    $\widehat{\boldsymbol{\beta}}(\lambda_r) \leftarrow \tilde{\boldsymbol{\beta}}_s(\lambda_r)$
    $\tilde{\boldsymbol{\beta}}_0(\lambda_{r+1}) \leftarrow \widehat{\boldsymbol{\beta}}(\lambda_r)$
end for

Task 3 - Outer Loop R Code

```r
LogisticLASSO <- function(dat, start, lambda) {
  r <- length(lambda)
  X <- as.matrix(cbind(rep(1, nrow(dat)), dat[, -1])) # design matrix
  y <- dat[, 1] # response vector
  res <- matrix(NA, nrow = r, ncol = ncol(dat) + 1)
  for (i in 1:r) {
    betavec <- MiddleLoop(X = X, y = y, start = start, lambda = lambda[i])
    res[i, ] <- c(lambda[i], betavec)
    start <- betavec
  }
  colnames(res) <- c("lambda", "(Intercept)", names(dat)[-1])
  return(res)
}
```

Task 3 - Middle Loop R Code

```r
MiddleLoop <- function(X, y, start, lambda, maxiter = 100) {
  betavec <- start
  u <- X %*% betavec
  p_vec <- sigmoid(u) # function `sigmoid` to compute exp(x)/(1 + exp(x))
  w <- p_vec * (1 - p_vec)
  eps <- 1e-5
  p_vec[p_vec < eps] <- 0
  p_vec[p_vec > 1 - eps] <- 1
  w[p_vec == 1 | p_vec == 0] <- eps
  z <- u + (y - p_vec) / w
  s <- 0
  t <- 2
  while (t > 1 && s < maxiter) { # if number of iterations of inner loop = 1, converge.
    s <- s + 1
    betavec <- InnerLoop(X = X, z = z, w = w, betavec = betavec, lambda = lambda)
    t <- betavec[1]
    betavec <- betavec[-1]
    u <- X %*% betavec
  }
  return(betavec)
}
```

Task 3 - Inner Loop R Code

```r
InnerLoop <- function(X, z, w, betavec, lambda, tol = 1e-10, maxiter = 1000) {
  prevfunc <- Inf
  curfunc <- coordinate_func(X = X, z = z, w = w, betavec = betavec, lambda = lambda)
  t <- 0
  while (abs(curfunc - prevfunc) > tol && t < maxiter) {
    t <- t + 1
    prevfunc <- curfunc
    betavec[1] <- sum(w * (z - X[, -1] %*% betavec[-1])) / sum(w)
    for (j in 2:length(betavec)) {
      betavec[j] <-
        soft.threshold(
          z = sum(w * X[, j] * (z - X[, -j] %*% betavec[-j])) / nrow(X),
          gamma = lambda
          ) *
        nrow(X) / sum(w * X[, j]^2)
    }
    curfunc <- coordinate_func(X = X, z = z, w = w, betavec = betavec, lambda = lambda)
  }
  return(c(t, betavec))
}
```

Task 4 - Objective

Objective:

Use 5-fold cross-validation to select the best λ. Compare the prediction performance between the 'optimal' model and 'full' model.

- Select the best λ: our function `cv.logit.lasso` and `cv.glmnet`.
- Use the best lambda to select predictors and refit logistic regression model –'optimal' model.
- Compare the 'optimal' model and the 'full' model from Task 2 in Specificity, Sensitivity, and AUC.

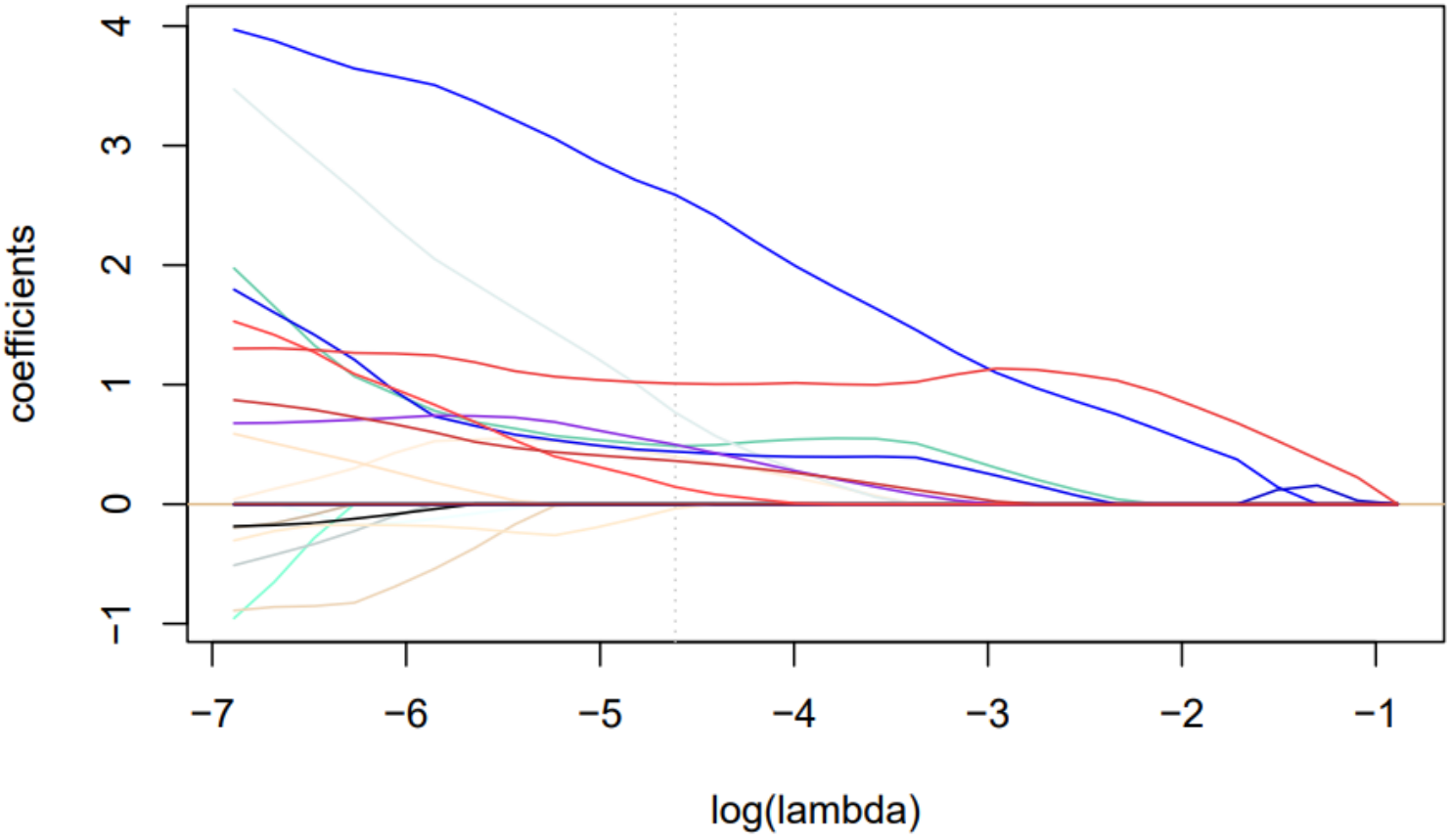
Task 4 - Our CV function

```
cv.logit.lasso <- function(x, y, nfolds = 5, lambda) {
  auc <- data.frame(matrix(ncol = 3, nrow = 0))
  folds <- createFolds(y, k = nfolds)
  for (i in 1:nfolds) {
    valid_index <- folds[[i]]
    x_training <- x[-valid_index, ]
    y_training <- y[-valid_index]
    training_dat <- data.frame(cbind(y_training, x_training))
    x_valid <- cbind(rep(1, length(valid_index)), x[valid_index, ])
    y_valid <- y[valid_index]
    res <- LogisticLASSO(dat = training_dat, start = rep(0, ncol(training_dat)), lambda = lambda)
    for (k in 1:nrow(res)) {
      betavec <- res[k, 2:ncol(res)]
      u_valid <- x_valid %*% betavec
      phat_valid <- sigmoid(u_valid)[, 1]
      roc <- roc(response = y_valid, predictor = phat_valid)
      auc <- rbind(auc, c(lambda[k], i, roc$auc[1]))
    }
  }
  colnames(auc) <- c("lambda", "fold", "auc")
  cv_res <- auc %>%
    group_by(lambda) %>%
    summarize(auc_mean = mean(auc)) %>%
    mutate(auc_ranking = min_rank(desc(auc_mean)))
  bestlambda <- min(cv_res$lambda[cv_res$auc_ranking == 1])
  return(cv_res)
}
```

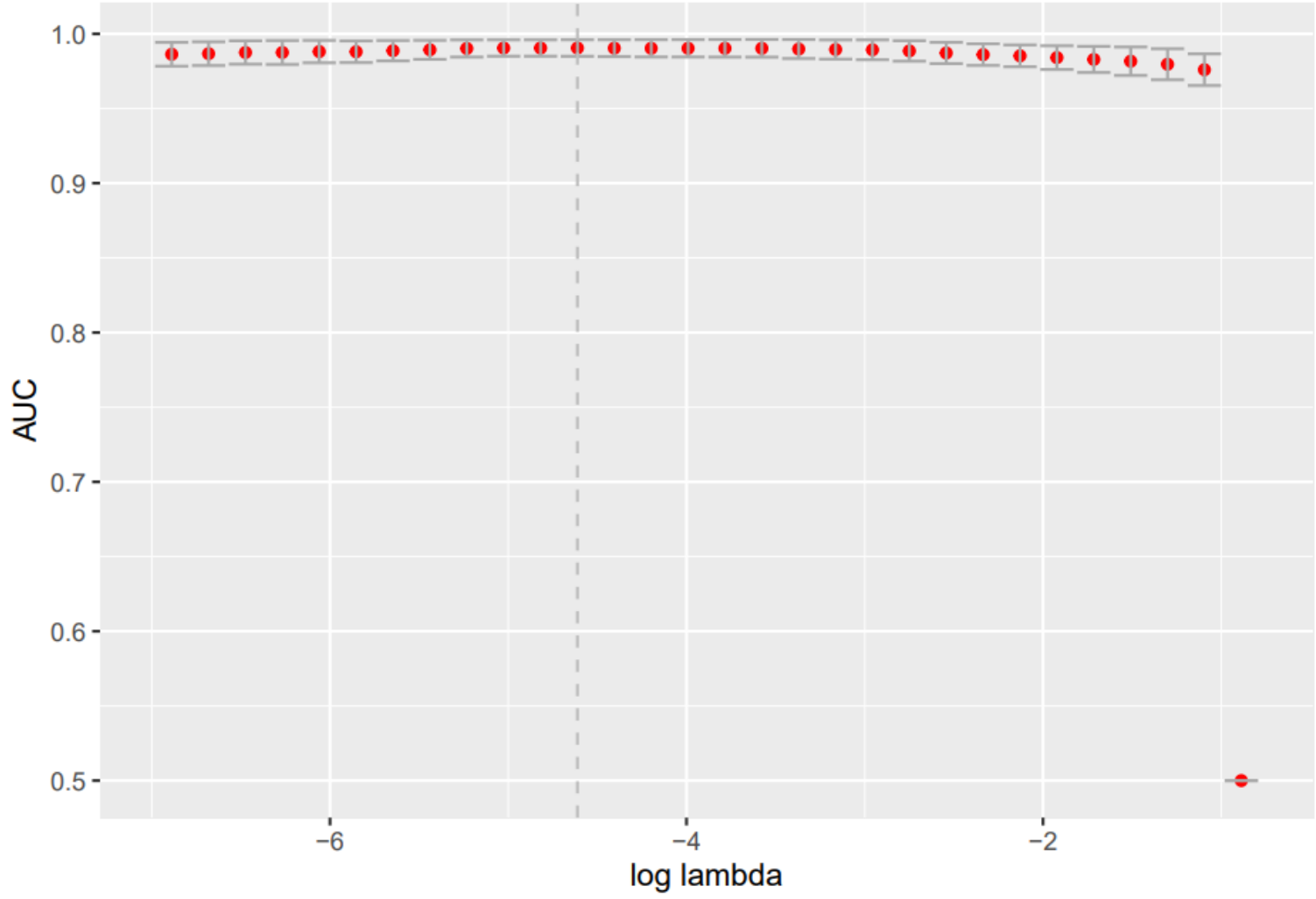
Task 4 - lambda Range

- λ_{max}: makes $\beta_k = 0$ for all $k \in \{1, 2, ..., p\}$.
- λ_{min}: λ_{max}/e^6.
- We use `seq` to produce 30 λ candidates, on a log scale.

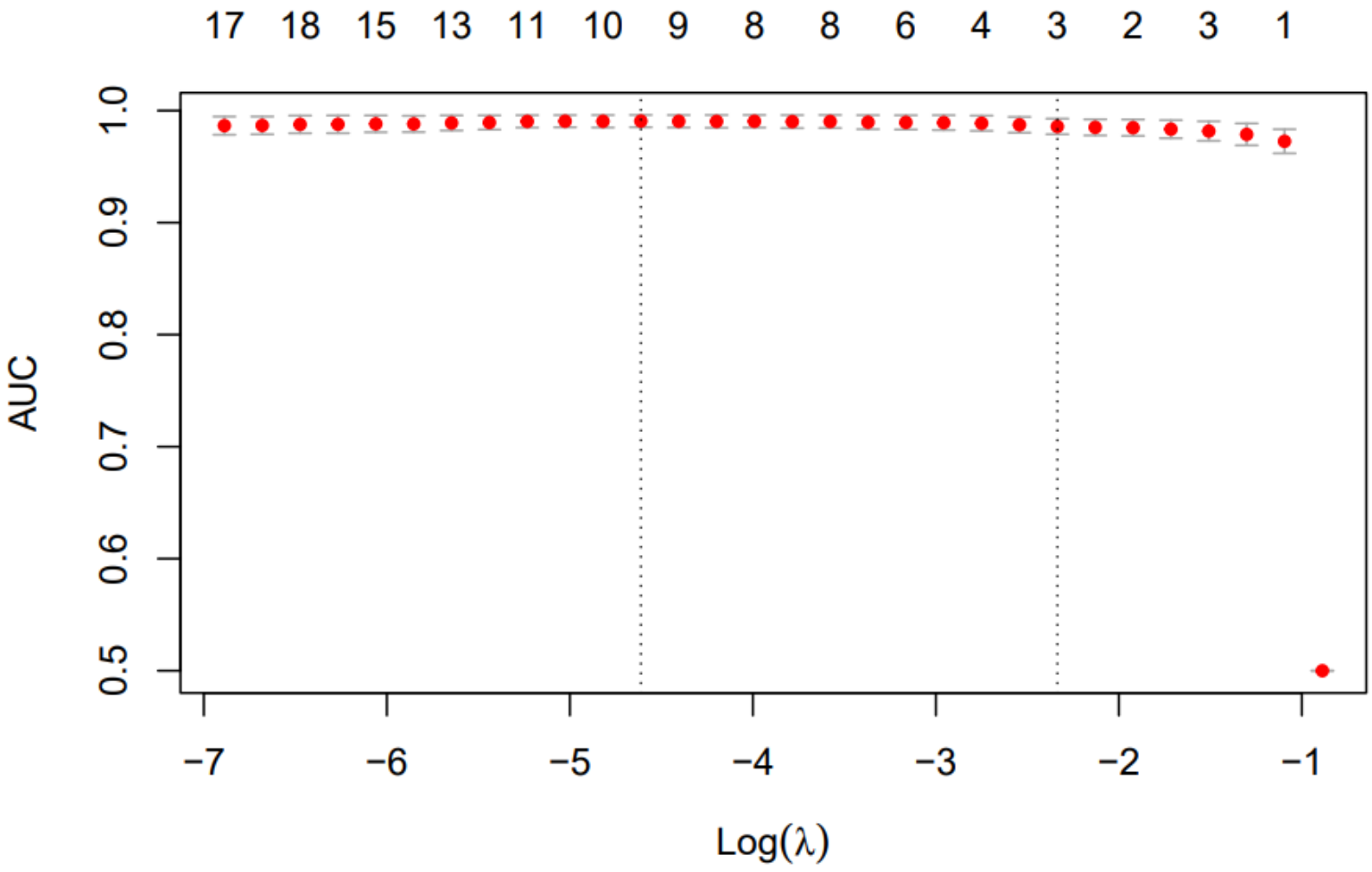
Task 4 - Our CV function – coefficient shrinkage



Task 4 - Our CV function – selecting the best lambda



Task 4 - glmnet – selecting the best lambda



Task 4 - Best lambda by the two methods

lambda	ours_AUC	cv.glmnet_AUC
0.4115445	0.5000000	0.5000000
0.3346284	0.9760054	0.9725915
0.2720876	0.9796678	0.9787526
0.2212355	0.9816903	0.9817053
0.1798874	0.9828852	0.9834002
0.1462671	0.9841071	0.9847216
0.1189303	0.9853029	0.9850192
0.0967027	0.9861031	0.9858166
0.0786293	0.9872060	0.9873215
0.0639338	0.9885875	0.9887026
0.0519848	0.9893408	0.9892575
0.0422691	0.9895397	0.9894566
0.0343691	0.9898394	0.9896556
0.0279457	0.9903360	0.9902522
0.0227227	0.9903354	0.9901527
0.0184759	0.9903346	0.9904508
0.0150229	0.9903347	0.9903518
0.0122151	0.9904332	0.9904501
0.0099322	0.9905325	0.9906491
0.0080759	0.9905317	0.9904501
0.0065665	0.9905324	0.9905479
0.0053393	0.9902345	0.9903485
0.0043414	0.9893390	0.9893569
0.0035300	0.9887423	0.9889618
0.0028703	0.9880447	0.9880654
0.0023338	0.9881480	0.9881674
0.0018976	0.9875518	0.9877693

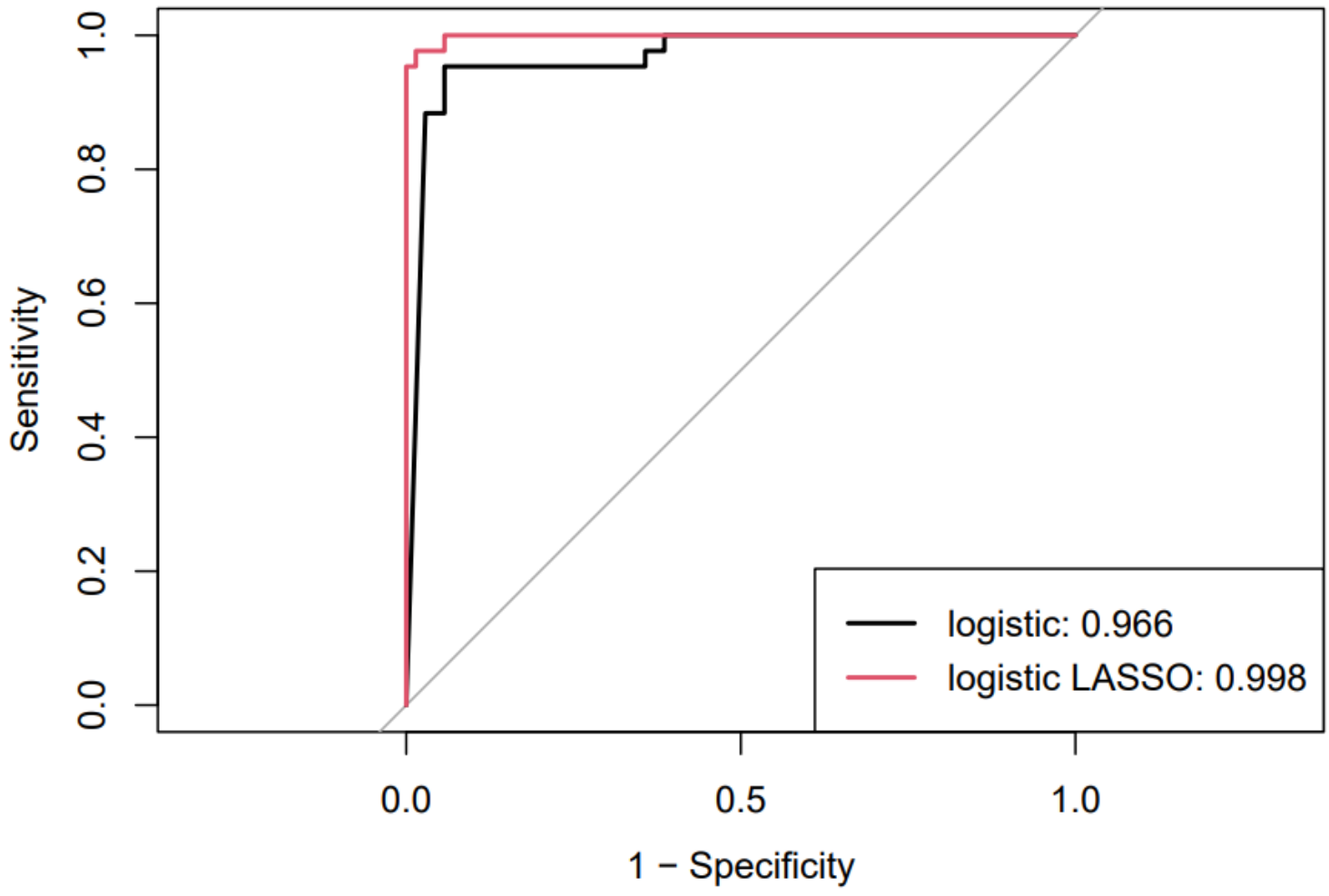
Task 4 - Selecting predictors

predictor	ours_coef	cv.glmnet_coef
(Intercept)	-0.6138280	-0.6048659
radius_mean	0.0000000	0.0000000
texture_mean	0.3958698	0.3811372
perimeter_mean	0.0000000	0.0000000
area_mean	0.0000000	0.0000000
smoothness_mean	0.0000000	0.0000000
compactness_mean	0.0000000	0.0000000
concavity_mean	0.0000000	0.0000000
concave.points_mean	0.4850029	0.4476499
symmetry_mean	0.0000000	0.0000000
fractal_dimension_mean	0.0000000	0.0000000
radius_se	0.7660262	0.8379136
texture_se	0.0000000	0.0000000
perimeter_se	0.0000000	0.0000000
area_se	0.0000000	0.0000000
smoothness_se	0.0000000	0.0000000
compactness_se	0.0000000	0.0000000
concavity_se	0.0000000	0.0000000
concave.points_se	0.0000000	0.0000000
symmetry_se	0.0000000	0.0000000
fractal_dimension_se	-0.0364481	-0.0368622
radius_worst	2.5878169	2.5734608
texture_worst	0.4394404	0.4590245
perimeter_worst	0.0000000	0.0000000
area_worst	0.0000000	0.0000000
smoothness_worst	0.4980005	0.4945458
compactness_worst	0.0000000	0.0000000
concavity_worst	0.1437060	0.1253231
concave.points_worst	1.0087922	1.0692685
symmetry_worst	0.3617036	0.3641634
fractal_dimension_worst	0.0000000	0.0000000

Task 4 - Compare 'optimal' and 'full' models - Table

Model	specificity	sensitivity	AUC
full	0.9268293	0.9305556	0.9661130
optimal	0.9545455	0.9855072	0.9983389

Task 4 - Compare ‘optimal’ and ‘full’ models - Plot



Discussions

- Lasso-logistic model performed better than the full logistic model.
- The most ideal performance is to classify each patient correctly, i.e. Specificity $=$ Sensitivity $= 1$
- Future work: Predictors need to be interpreted or have some important clinical significance.

Reference

Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. J Stat Softw. 2010;33(1):1-22. PMID: 20808728; PMCID: PMC2929880.

Q&A

- Thanks for listening!