P8160 - Breast Cancer Diagnosis

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

4/7/2023

Contents

1 Background and Objectives

Mammography is recognized as the most effective screening method for early breast cancer detection, but its accuracy remains limited. And as the number of variables that help predict breast cancer increases, doctors are forced to rely more on their subjective experiences to make decisions. The use of computer models can detect abnormalities in mammograms to aid radiologists in breast cancer diagnosis (Freer et al. 2001). The purpose of this study was to predict breast cancer benign/malignant status by quantitative modeling based on logistic regression, which may help radiologists manage a large amount of available information, make effective decisions to detect breast cancers and reduce unnecessary biopsy.

The data given has 569 observations. The column 'Diagnosis' which identifies if the image is coming from cancer tissue or benign cases (M = malignant, B = benign) would be used as outcome for modeling. We denote malignant as 1 and benign cases as 0 for prediction. The other 30 columns correspond to mean, standard deviation and the largest values (points on the tails) of the distributions of the following 10 features computed for the cell nuclei:

- radius: mean of distances from center to points on the perimeter
- texture: standard deviation of gray-scale values
- perimeter: mean size of the core tumor
- area: mean area of the core tumor
- smoothness: local variation in radius lengths
- compactness: perimeter^2/area - 1.0
- concavity: severity of concave portions of the contour
- concave points: number of concave portions of the contour
- symmetry: symmetry of the tumor
- fractal dimension: "coastline approximation" - 1

We partitioned the data into training data (80%) and test data (20%). As exploring the training data, there are many predictors highly correlated to each other as shown in Figure 1, such as 'area_mean', 'perimeter_worst', 'radius_mean' and so on, that could cause unstable parameter estimation as well as perplexing the interpretation of logistic model. While we could eliminate some correlated features for modeling as shown in Figure 2, regularized logistic regression also helps to tackle with it, which would be further explored in the following parts.

2 Methods

2.1 Logistic Model

Logistic model measures the probability of an event taking place by having the log-odds for the event be a linear combination of one or more independent variables, and is commonly used in classifying binary response variables.

Hereby, the variable "Diagnosis" is a binary response variable indicating if the image is coming from cancer tissue or benign cases (M = malignant, B = benign). In the following logistic regression model, the "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

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

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^\top \in \mathbb{R}^{p+1}$ is the parameter vector, $\mathbf{x}_i = (1, X_{i1}, \dots, X_{ip})^\top$ 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. Let $\mathbf{y} = (Y_1, Y_2, \dots, Y_n)^\top$ denote the response vector, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times (p+1)}$ denote the design matrix. The observed likelihood of $\{(Y_1, \mathbf{x}_1), (Y_2, \mathbf{x}_2) \dots, (Y_n, \mathbf{x}_n)\}$ is

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

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

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

The estimates of model parameters are

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

and the optimization problem is

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

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

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

Denote $w_i = p_i(1 - p_i) \in (0, \frac{1}{4}]$ and $\mathbf{W} = \mathrm{diag}(w_1, \dots, w_n)$. The Hessian matrix of f is given by

$$\begin{aligned} \nabla^2 f(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) &= -\mathbf{X}^\top \mathbf{W} \mathbf{X} \qquad (4) \\ &= -\sum_{i=1}^{n} w_i \mathbf{x}_i \mathbf{x}_i^\top \\ &= -\begin{pmatrix} \sum_{i=1}^{n} w_i & \sum_{i=1}^{n} w_i X_{i1} & \cdots & \sum_{i=1}^{n} w_i X_{i1} \\ \sum_{i=1}^{n} w_i X_{i1} & \sum_{i=1}^{n} w_i X_{i1}^2 & \cdots & \sum_{i=1}^{n} w_i X_{i1} X_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} w_i X_{ip} & \sum_{i=1}^{n} w_i X_{ip} X_{i1} & \cdots & \sum_{i=1}^{n} w_i X_{ip}^2 \end{pmatrix}. \end{aligned}$$

It can be shown that the Hessian matrix $\nabla^2 f(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})$ is a negative-definite matrix if $\mathbf{X}$ has full rank (Theorem 1). Hence, the optimization problem (3) is a well-defined problem.

2.2 Newton-Raphson Algorithm

2.2.1 Algorithm Design

Given that the derivative of the log-likelihood function with respect to each parameter is nonlinear and difficult to solve analytically for maximum likelihood, we use the Newton-Raphson algorithm to solve the optimization problem (3) numerically.

We develop a modified Newton-Raphson algorithm including a step-halving step. Given that the Hessian matrix is negative-definite in this case, we don't need to ensure that the direction of the step is an ascent direction.

Algorithm 1 Newton-Raphson algorithm including a step-halving step

```
Require: $f(\boldsymbol{\beta})$ - target function as given in (2); $\boldsymbol{\beta}_0$ - starting value
Ensure: $\widehat{\boldsymbol{\beta}}$ such that $\widehat{\boldsymbol{\beta}} \approx \arg\max_{\boldsymbol{\beta}} \; f(\boldsymbol{\beta})$
  $i \leftarrow 0$, where $i$ is the current number of iterations
  $f(\boldsymbol{\beta}_{-1}) \leftarrow -\infty$
  while convergence criterion is not met do
    $i \leftarrow i + 1$
    $\mathbf{d}_i \leftarrow -[\nabla^2 f(\boldsymbol{\beta}_{i-1})]^{-1} \nabla f(\boldsymbol{\beta}_{i-1})$, where $\mathbf{d}_i$ is the direction in the $i$-th iteration
    $\lambda_i \leftarrow 1$, where $\lambda_i$ is the multiplier in the $i$-th iteration
    $\boldsymbol{\beta}_i \leftarrow \boldsymbol{\beta}_{i-1} + \lambda_i \mathbf{d}_i$
    while $f(\boldsymbol{\beta}_i) \le f(\boldsymbol{\beta}_{i-1})$ do
      $\lambda_i \leftarrow \lambda_i / 2$
      $\boldsymbol{\beta}_i \leftarrow \boldsymbol{\beta}_{i-1} + \lambda_i \mathbf{d}_i$
    end while
  end while
  $\widehat{\boldsymbol{\beta}} \leftarrow \boldsymbol{\beta}_i$
```

We implement the algorithm in R (see Code 1). The starting values of $\boldsymbol{\beta}$ is set to all 0's.

2.2.2 Complete Separation Problem

However, the algorithm does not converge when a complete separation occurs. A complete separation in a logistic regression, also referred to as perfect prediction, occurs whenever there exists some vector of coefficients $\hat{\boldsymbol{\beta}}$ such that $Y_i = 1$ whenever $\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} > 0$ and $Y_i = 0$ whenever $\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} \le 0$. In other words, complete separation occurs whenever a linear function of predictors can generate perfect predictions of response.

To further explain this problem, we prove that: if there exists a vector of coefficients $\hat{\boldsymbol{\beta}}$ that can generate perfect predictions, there does not exist $\boldsymbol{\beta}^* \in \mathbb{R}^{p+1}$ such that $\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} f(\boldsymbol{\beta})$, where f is given in (2) (see Theorem 2). Thus our Newton-Raphson algorithm does not converge.

2.3 Logistic-LASSO Model

Regularization is the common approach for variable selection, in which LASSO is to add L-1 penalty to the objective function. In the context of logistic regression, LASSO estimates the model parameters $\boldsymbol{\beta}$ by optimizing a penalized loss function:

$$\min_{\boldsymbol{\beta}} \; -\frac{1}{n} f(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|. \tag{5}$$

where $\lambda \ge 0$ is the tuning parameter and f is the log-likelihood function given in (2). Note that the intercept is not penalized and all predictors are standardized.

Then we could develop a path-wise coordinate descent algorithm to solve the optimization problem (5) with a sequence of nested loops:

Algorithm 2 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 (2); $\boldsymbol{\beta}_0$ - starting value; $\{\lambda_1, \ldots, \lambda_m\}$ - a sequence of descending $\lambda$'s, where $\lambda_1 = \lambda_{max}$ is given in (6); $\epsilon$ - 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 (7) 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
```

Outer Loop. In the outer loop, we compute the solutions of the optimization problem (5) for a decreasing sequence of values for λ: $\{\lambda_1, \ldots, \lambda_m\}$, starting at the smallest value $\lambda_1 = \lambda_{max}$ for which the estimates of all coefficients $\hat{\beta}_j = 0, \ j = 1, 2, \ldots, p$, which is

$$\lambda_{max} = \max_{j \in \{1, \ldots, p\}} \left| \frac{1}{n} \sum_{i=1}^{n} X_{ij} (Y_i - \bar{Y}) \right|, \tag{6}$$

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). Apart from giving us a path of solutions, this scheme exploits warm starts, and leads to a more stable algorithm.

Middle Loop. In the middle loop, we find the estimates of $\boldsymbol{\beta}$ by solving the optimization problem (5) for a fixed λ. For each iteration of the middle loop, based on the current parameter estimates $\tilde{\boldsymbol{\beta}}$, we form a quadratic approximation to the log-likelihood f using a Taylor expansion:

$$\begin{aligned} f(\boldsymbol{\beta}) \approx \ell(\boldsymbol{\beta}) &= f(\tilde{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})^\top \nabla f(\tilde{\boldsymbol{\beta}}) + \frac{1}{2} (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})^\top \nabla^2 f(\tilde{\boldsymbol{\beta}}) (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) \\ &= f(\tilde{\boldsymbol{\beta}}) + [\mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})]^\top (\mathbf{y} - \tilde{\mathbf{p}}) - \frac{1}{2} [\mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})]^\top \tilde{\mathbf{W}} \mathbf{X} (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) \end{aligned}$$

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

where $\tilde{\mathbf{p}} = (\tilde{p}_1, \ldots, \tilde{p}_n)^\top$ and $\tilde{\mathbf{W}} = \operatorname{diag}(\tilde{w}_1, \ldots, \tilde{w}_n)$ are the estimates of $\mathbf{p}$ and $\mathbf{W}$ based on $\tilde{\boldsymbol{\beta}}$. We rewrite the function $\ell(\boldsymbol{\beta})$ as follows:

$$
\ell(\boldsymbol{\beta}) = -\frac{1}{2}\sum_{i=1}^{n}\tilde{w}_i(\tilde{z}_i - \mathbf{x}_i^\top\boldsymbol{\beta})^2 + C(\tilde{\boldsymbol{\beta}}), \tag{7}
$$

where

$$
\tilde{z}_i = \mathbf{x}_i^\top\tilde{\boldsymbol{\beta}} + \frac{Y_i - \tilde{p}_i}{\tilde{w}_i}
$$

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

Inner Loop. In the inner loop, with fixed $\tilde{w}_i$'s, $\tilde{z}_i$'s, and thus a fixed form of ℓ based on the estimates of $\boldsymbol{\beta}$ in the previous iteration of the middle loop, we update the estimates of $\boldsymbol{\beta}$ by solving a modified optimization problem of (5):

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

which is equivalent to the following penalized weighted least-squares problem

$$
\min_{\boldsymbol{\beta}} \; \frac{1}{2n}\sum_{i=1}^{n}\tilde{w}_i(\tilde{z}_i - \mathbf{x}_i^\top\boldsymbol{\beta})^2 + \lambda\sum_{k=1}^{p}|\beta_k|. \tag{8}
$$

We use coordinate descent to update the estimates of $\boldsymbol{\beta}$. For each iteration of the inner loop, suppose we have the current estimates $\tilde{\beta}_k$ for $k \neq j$ and we wish to partially optimize with respect to β_j:

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

Thus the coordinate descent has updates

$$
\begin{aligned}
\tilde{\beta}_0 &\leftarrow \frac{\sum_{i=1}^{n}\tilde{w}_i(\tilde{z}_i - \sum_{k=1}^{p}X_{ik}\tilde{\beta}_k)}{\sum_{i=1}^{n}\tilde{w}_i}, \\
\tilde{\beta}_j &\leftarrow \frac{S\left(\frac{1}{n}\sum_{i=1}^{n}\tilde{w}_iX_{ij}(\tilde{z}_i - \sum_{k\neq j}X_{ik}\tilde{\beta}_k), \lambda\right)}{\frac{1}{n}\sum_{i=1}^{n}\tilde{w}_iX_{ij}^2}, \; j = 1, \ldots, p
\end{aligned}
$$

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

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

We can then update estimates of β_j's repeatedly for $j = 0, 1, 2, ..., p, 0, 1, 2, ...$ until convergence.

We implement the algorithm in R (see Code 2). Here are some important details regarding the implementation of the algorithm:

- In the outer loop, the starting values of $\boldsymbol{\beta}$ is set to all 0's. Note that the value of λ_{max} is different from the form given in Friedman et al. (2010), which is $\lambda_{max} = \max_j |\langle \mathbf{x}_{\cdot j}, \mathbf{y} \rangle|$, where $\mathbf{x}_{\cdot j}$ is the j-th column of the design matrix $\mathbf{X}$, for $j = 1, \dots, p$. This is because we also take the update of $\tilde{\beta}_0$ into consideration in the inner loop. (see Theorem 3 in the appendices) In practice, we add a very small value (e.g., 10^{-10}) to λ_{max} to avoid computational errors that may cause one slope coefficient estimate not equal to zero.
- In the middle loop, care is taken to avoid coefficients diverging in order to achieve fitted probabilities of 0 or 1. When $\tilde{p}_i$ is within $\delta = 10^{-5}$ of 1, we set it to 1, and set the corresponding weight $\tilde{w_i}$ to δ. 0 is treated similarly.
- The stopping criteria of the middle loop is either the number of iterations of the inner loop is exactly 1 or it reaches a maximum number of iterations 100. The stopping criteria of the inner loop is either the difference of the penalized loss function given in (8) between the two iterations is less than the tolerance $\epsilon = 10^{-10}$ or it reaches a maximum number of iterations 1000.

2.4 Five-fold Cross-validation for LASSO

Since the Logistic-Lasso model depends on penalty term for variable selection, we further develop 5-fold cross-validation to select the tuning parameter λ to obtain the optimized result.

For the initial λ range, we use a sequence of descending λ's, starting with the largest λ_1, which is the maximum value of λ_{max} in each training set as given in (6). We set $\lambda_{min} = \lambda_1/e^6$, and create 30 λ candidates on a log scale.

To select the optimal tuning parameter λ, the original data is randomly shuffled and split into five equally sized groups. One of the groups is used as the validation set, while the remaining groups are used as the training set. The path-wise coordinate-wise optimization algorithm is then applied to the training set, and AUC scores are calculated for each λ using the validation set. This process is repeated until each of the five groups has been used as the validation set, and the mean AUC for each λ is computed.

We select the best λ in two ways. One way is that we wrote a function to do the 5-fold cross-validation (see Code 3) and the other way is to use the `cv.glmnet` function from the popular package `caret`.

Here we take the λ with the greatest mean AUC as the best λ. Further, we look at the predictors that the best λ chooses and take them to re-fit the logistic regression model on the training data which is the 'optimal' model. We also fit a logsitical regression model on the training data using our Newton-Raphson algorithm to get the 'full' model. After that, we compare the two models' prediction performance on the test data in specificity, sensitivity, and AUC.

3 Results

3.1 Five-fold CV for Logistic-LASSO

Both our cross-validation function and `cv.glmnet` function reach the greatest mean AUC when $\lambda = 0.0099322$, thus this is the best λ we choose. As for the predictors, when $\lambda = \lambda_{best}$, both methods select ten same predictors, which are `texture_mean`, `concave.points_mean`, `radius_se`, `fractal_dimension_se`, `radius_worst`, `texture_worst`, `smoothness_worst`, `concavity_worst`, `concave.points_worst`, and `symmetry_worst`.

3.2 Model Comparison

The ten selected predictors are used to re-fit the logistic regression model as the 'optimal' model, which is then compared with the 'full' model generated by the Newton-Raphson algorithm. The result turns out that the two models differ in prediction performance. The ‘optimal’ model outperformed the ‘full’ model with higher specificity, sensitivity, larger AUC, and fewer predictors.

4 Discussion

The main conclusion is that the logistic Lasso model performed better than the full logistic model in this case. Recall that our goal is to build a predictive model to classify each patient’s cancer diagnosis accurately according to the information extracted from the images. The specificity and sensitivity should be as high as possible.

For future work, now that we have selected the ten predictors, we want to know why do these ten matter. What’s the interpretation of this model? Do these predictors have any clinical significance? These questions may need to be answered by a cancer expert.

Group Contributions

References

Freer, Timothy W., and Michael J. Ulissey. "Screening mammography with computer-aided detection: prospective study of 12,860 patients in a community breast center." Radiology 220.3 (2001): 781-786.

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.

Appendices

Figures and Tables

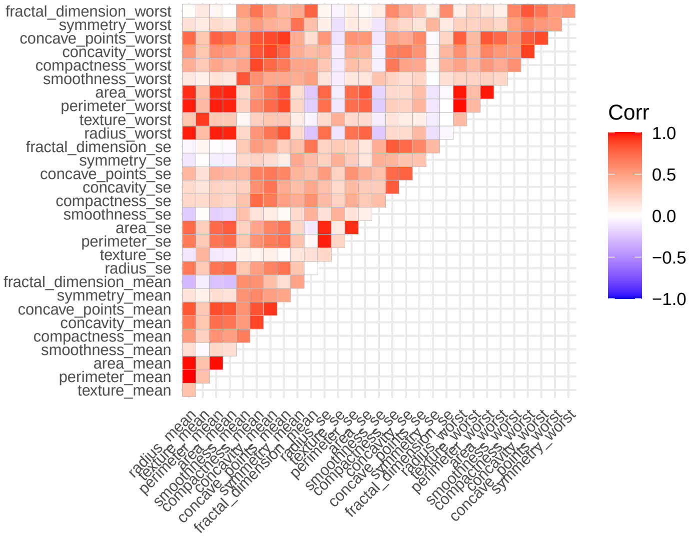


Figure 1: Variables Correlation Part I

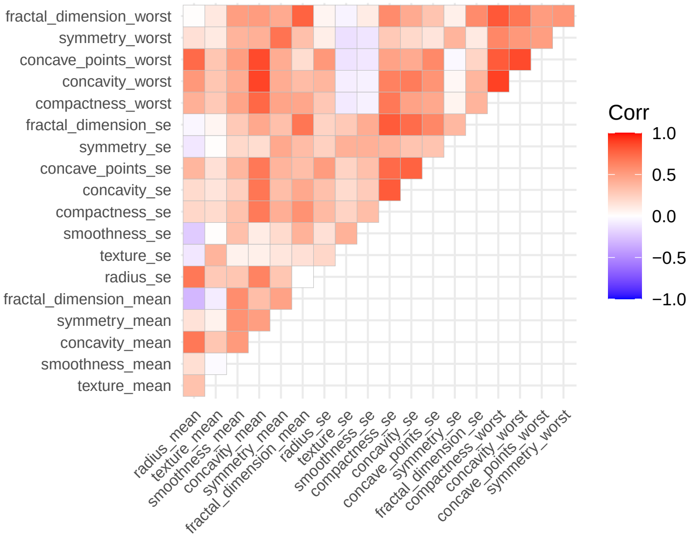


Figure 2: Variables Correlation Part II

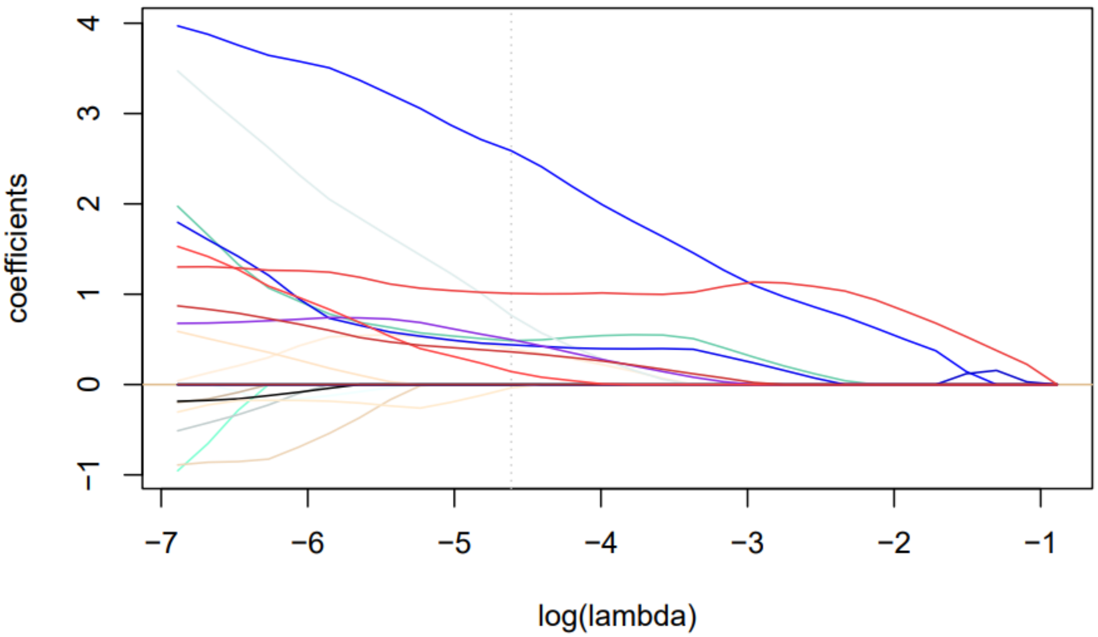


Figure 3: LASSO coefficients shrinkage using our function

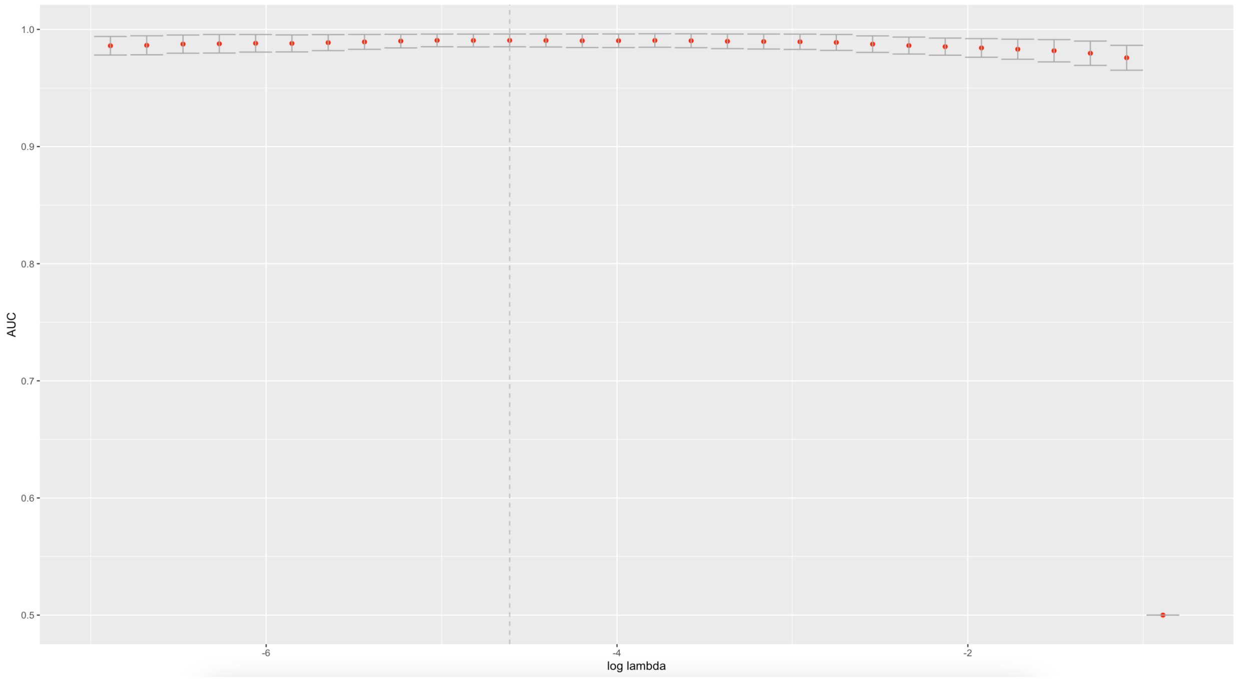
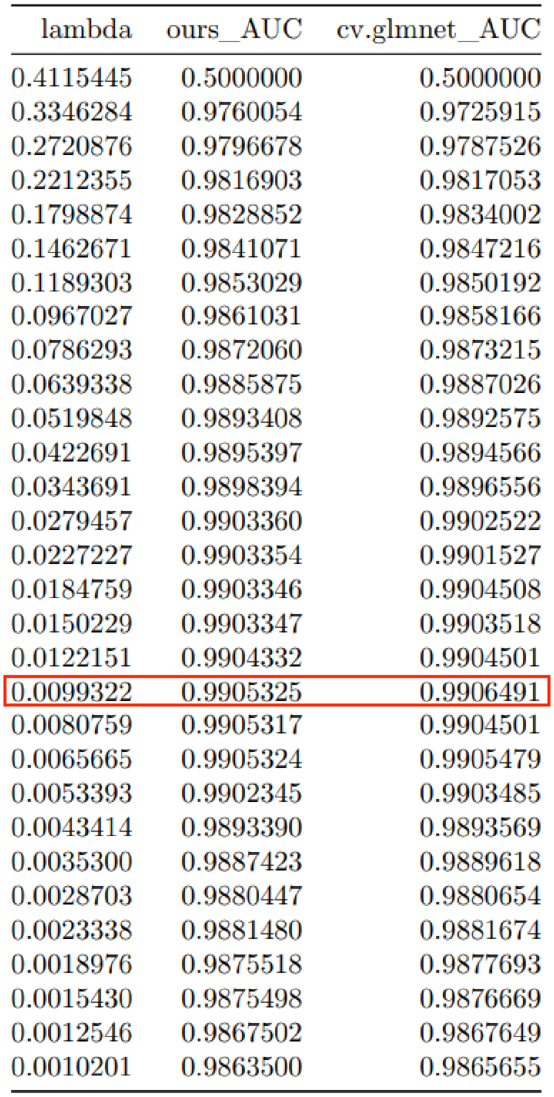
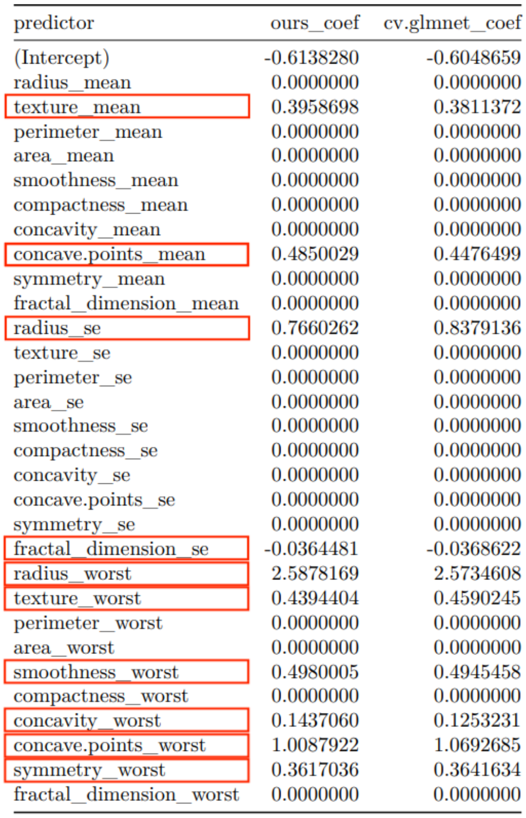


Figure 4: 5-fold CV plot for varying values of tuning paramter λ using our function



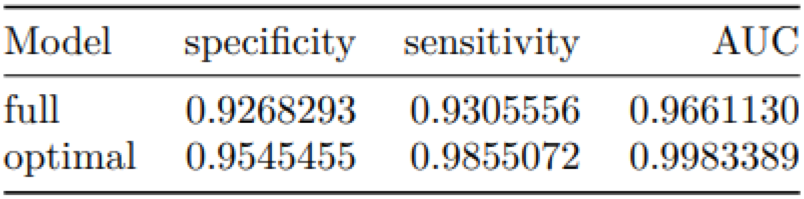
lambda	ours_AUC	cv.glmnet_AUC
0.4115445	0.5000000	0.5000000
0.3346284	0.9760054	0.9725915
0.2720876	0.9796678	0.9787526
0.2212355	0.9816903	0.9817053
0.1798874	0.9828852	0.9834002
0.1462671	0.9841071	0.9847216
0.1189303	0.9853029	0.9850192
0.0967027	0.9861031	0.9858166
0.0786293	0.9872060	0.9873215
0.0639338	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
0.0015430	0.9875498	0.9876669
0.0012546	0.9867502	0.9867649
0.0010201	0.9863500	0.9865655

Figure 5: 5-fold CV results for varying values of tuning paramter λ using our function and ‘cv.glmnet‘



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

Figure 6: Selected predictors by LASSO using our function and ‘cv.glmnet‘



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

Figure 7: Comparison of the two models’ performance on the test dataset

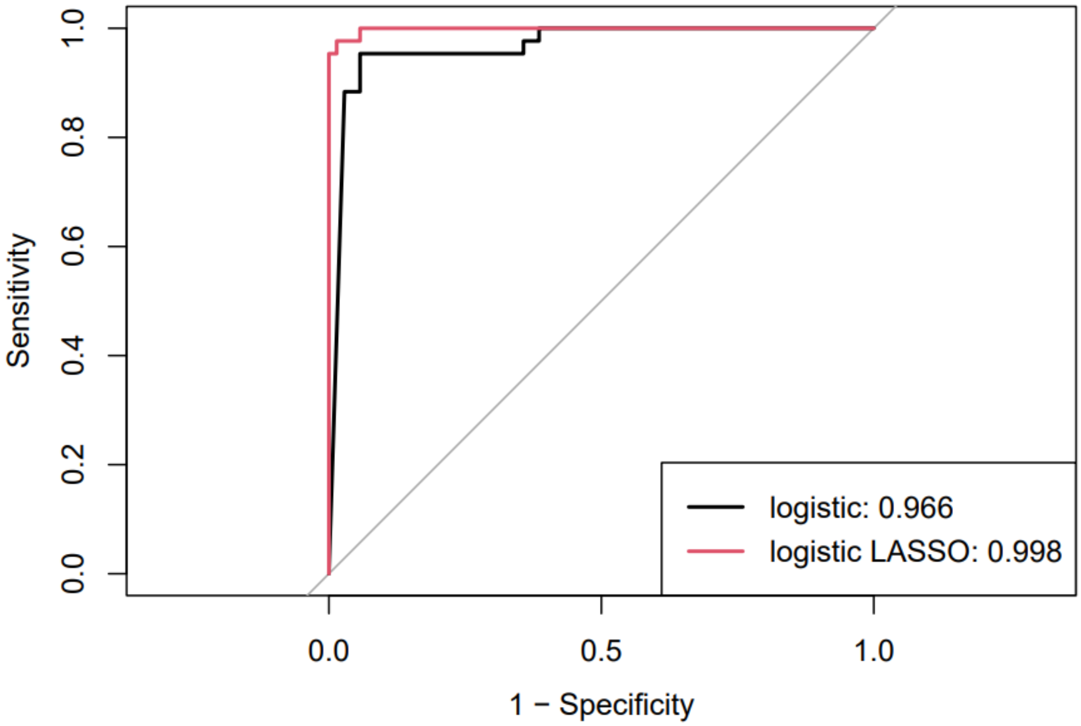


Figure 8: ROC curve of the two models on the test dataset

R functions

```
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)
}
```

Code 1: Newton-Raphson Algorithm

```
# outer loop
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)
}

# middle loop

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

# inner loop
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))
}
```

Code 2: Logistic Lasso Algorithm

```
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$auc[1]))
    }
  }
  colnames(auc) <- c("lambda", "fold", "auc")
  cv_res <- auc %>%
    group_by(lambda) %>%
    summarize(auc_mean = mean(auc),
              auc_se = sd(auc) / sqrt(nfolds),
              auc_low = auc_mean - auc_se,
              auc_high = auc_mean + auc_se) %>%
    mutate(auc_ranking = min_rank(desc(auc_mean)))
  bestlambda <- max(cv_res$lambda[cv_res$auc_ranking == 1])
  return(cv_res)
}
```

Code 3: 5-fold cross-validation to select best lambda

Proofs

Theorem 1. *The Hessian matrix $\nabla^2 f(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X})$ as given in (4) 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

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

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

Theorem 2. *If there exists a vector of coefficients $\hat{\boldsymbol{\beta}}$ such that $Y_i = 1$ whenever $\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} > 0$ and $Y_i = 0$ whenever $\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} \le 0$, there does not exist $\boldsymbol{\beta}^* \in \mathbb{R}^{p+1}$ such that $\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} f(\boldsymbol{\beta})$, where f is given in (2).*

Proof. Assume such $\boldsymbol{\beta}^*$ exists, then $\forall \boldsymbol{\beta} \in \mathbb{R}^{p+1}$, we have $f(\boldsymbol{\beta}) \le f(\boldsymbol{\beta}^*)$.

First, we prove that: there exists a vector of coefficients $\tilde{\boldsymbol{\beta}}$ such that $Y_i = 1$ whenever $\mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} > 0$ and $Y_i = 0$ whenever $\mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} < 0$.

Let $A_1 = \{i : Y_i = 1\} = \{i : \mathbf{x}_i^\top \hat{\boldsymbol{\beta}} > 0\}$ and $A_0 = \{i : Y_i = 0\} = \{i : \mathbf{x}_i^\top \hat{\boldsymbol{\beta}} \le 0\}$. Then we have

$$\epsilon := \min_{i \in A_1}(\mathbf{x}_i^\top \hat{\boldsymbol{\beta}}) > 0.$$

Let $\tilde{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} - (\epsilon/2, 0, \ldots, 0)^\top$. Given that $X_{i0} = 1$ for all i, we have

$$\begin{aligned}\mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} &= \mathbf{x}_i^\top \hat{\boldsymbol{\beta}} - \epsilon/2 \cdot 1 \ge \epsilon - \epsilon/2 = \epsilon/2 > 0, \quad \forall i \in A_1 \\ \mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} &= \mathbf{x}_i^\top \hat{\boldsymbol{\beta}} - \epsilon/2 \cdot 1 \le 0 - \epsilon/2 = -\epsilon/2 < 0, \quad \forall i \in A_0\end{aligned}$$

Thus we have $A_1 = \{i : Y_i = 1\} = \{i : \mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} > 0\}$ and $A_0 = \{i : Y_i = 0\} = \{i : \mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} < 0\}$.

Next, we prove that

$$\lim_{k \to \infty} f(k\tilde{\boldsymbol{\beta}}) = 0. \tag{9}$$

$\forall k > 0$, we have $A_1 = \{i : \mathbf{x}_i^\top (k\tilde{\boldsymbol{\beta}}) > 0\}$ and $A_0 = \{i : \mathbf{x}_i^\top (k\tilde{\boldsymbol{\beta}}) < 0\}$.

Thus,

$$\begin{aligned}\lim_{k \to \infty} f(k\tilde{\boldsymbol{\beta}}) &= \lim_{k \to \infty} \sum_{i \in A_1} \left[Y_i \mathbf{x}_i^\top (k\tilde{\boldsymbol{\beta}}) - \log\left(1 + e^{\mathbf{x}_i^\top (k\tilde{\boldsymbol{\beta}})}\right)\right] + \lim_{k \to \infty} \sum_{i \in A_0} \left[Y_i \mathbf{x}_i^\top (k\tilde{\boldsymbol{\beta}}) - \log\left(1 + e^{\mathbf{x}_i^\top (k\tilde{\boldsymbol{\beta}})}\right)\right] \\ &= \sum_{i \in A_1} \lim_{k \to \infty} \left[k\mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} - \log\left(1 + e^{k\mathbf{x}_i^\top \tilde{\boldsymbol{\beta}}}\right)\right] + \sum_{i \in A_0} \lim_{k \to \infty} \left[-\log\left(1 + e^{k\mathbf{x}_i^\top \tilde{\boldsymbol{\beta}}}\right)\right] \\ &= \sum_{i \in A_1} \lim_{z \to \infty} \left[z - \log\left(1 + e^z\right)\right] + \sum_{i \in A_0} (-\log 1) \\ &= 0 + 0 = 0.\end{aligned}$$

Last, we prove that: there exists $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ such that $f(\boldsymbol{\beta}) > f(\boldsymbol{\beta}^*)$, which is contradictory to the statement that $\forall \boldsymbol{\beta} \in \mathbb{R}^{p+1}$, $f(\boldsymbol{\beta}) \le f(\boldsymbol{\beta}^*)$.

Notice that $f(\boldsymbol{\beta}) < 0$ holds for any $\boldsymbol{\beta} \in \mathbb{R}$, then we have $f(\boldsymbol{\beta}^*) < 0$.

Given that $f(\boldsymbol{\beta}^*) < 0$ and (9) holds, there exists $N \in \mathbb{R}$ such that $\forall k > N$, $f(k\tilde{\boldsymbol{\beta}}) > f(\boldsymbol{\beta}^*)$.

Thus our assumption must be false. □

Theorem 3. *Given that the starting values of $\boldsymbol{\beta}$ of the path-wise coordinate-wise descent algorithm are all 0's, and the coordinate descent in the inner loop has updates*

$$\begin{aligned}\tilde{\beta}_0 &\leftarrow \frac{\sum_{i=1}^n \tilde{w}_i(\tilde{z}_i - \sum_{k=1}^p X_{ik}\tilde{\beta}_k)}{\sum_{i=1}^n \tilde{w}_i},\\ \tilde{\beta}_j &\leftarrow \frac{S\left(\frac{1}{n}\sum_{i=1}^n \tilde{w}_i X_{ij}(\tilde{z}_i - \sum_{k\neq j} X_{ik}\tilde{\beta}_k), \lambda\right)}{\frac{1}{n}\sum_{i=1}^n \tilde{w}_i X_{ij}^2},\ j = 1, \ldots, p\end{aligned}$$

the smallest λ for which the updates of all slope coefficient estimates $\tilde{\beta}_j = 0,\ j = 1, 2, \ldots, p$ in every iteration of the inner loop is given by

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

where $\bar{Y} = \frac{1}{n}\sum_{i=1}^n Y_i$.

Proof. Denote $\tilde{\boldsymbol{\beta}}^{(s,t)}$ as the current estimates of $\boldsymbol{\beta}$ in the first outer loop after the s-th iteration of the middle loop and the t-th iteration of the inner loop, $s, t \in \{0, 1, \ldots\}$. Then $\tilde{\boldsymbol{\beta}}^{(0,0)} = \mathbf{0}$ denotes the starting values, and $\tilde{\boldsymbol{\beta}}^{(s-1,N_s)} = \tilde{\boldsymbol{\beta}}^{(s,0)}$, where N_s is the total number of iterations of inner loop in the s-th middle loop. We want to prove that λ_{max} as given in the theorem is the smallest value that can ensure that $\forall s, t$, $\tilde{\beta}_j^{(s,t)} = 0$, $j = 1, \ldots, p$.

We prove it by induction. Assume that $\forall k_t < N_{k_s}$ when $k_s < s$ and $\forall k_t < t$ when $k_s = s$, we have $\tilde{\beta}_j^{(k_s,k_t)} = 0$, $j = 1, \ldots, p$, where $t \leq N_s$. We want to find the smallest λ such that $\tilde{\beta}_j^{(s,t)} = 0$, $j = 1, \ldots, p$.

Let $\tilde{z}_i^{(s)}$, $\tilde{p}_i^{(s)}$, and $\tilde{w}_i^{(s)}$ denote the updates of z_i, p_i, and w_i in the ℓ function in the s-th middle loop as given in (7). Then we have

$$\tilde{p}_i^{(s)} = \frac{e^{\tilde{\beta}_0^{(s,0)}}}{1 + e^{\tilde{\beta}_0^{(s,0)}}}$$

is a constant value for all i's. Denote this value as $\tilde{p}^{(s)}$, then $\forall i = 1, \ldots, n$, we have

$$\begin{aligned}\tilde{w}^{(s)} := \tilde{w}_i^{(s)} &= \tilde{p}^{(s)}(1 - \tilde{p}^{(s)}),\\ \tilde{z}_i^{(s)} &= \tilde{\beta}_0^{(s,0)} + \frac{Y_i - \tilde{p}^{(s)}}{\tilde{w}^{(s)}}.\end{aligned}$$

Then we have

$$\tilde{\beta}_0^{(s,k_t)} = \frac{\sum_{i=1}^n \tilde{w}^{(s)}\tilde{z}_i}{n\tilde{w}^{(s)}} = \tilde{\beta}_0^{(s,0)} + \frac{\bar{Y} - \tilde{p}^{(s)}}{\tilde{w}^{(s)}},\ k_t = 1, 2, \ldots, t$$

Since the soft-thresholding operator $S(z,\gamma) = 0$ if and only if $\gamma \geq |z|$, $\tilde{\beta}_j^{(s,t)} = 0,\ j = 1, \ldots, p$ if and only if

$$\frac{1}{n}\left|\sum_{i=1}^n \tilde{w}^{(s)} X_{ij}\left(\tilde{z}_i^{(s)} - \sum_{k<j} X_{ik}\tilde{\beta}_k^{(s,t)} - \sum_{k>j} X_{ik}\tilde{\beta}_k^{(s,t-1)}\right)\right| \leq \lambda,\ j = 1, \ldots, p$$

Since we have

$$\begin{aligned}\frac{1}{n}\left|\sum_{i=1}^n \tilde{w}^{(s)} X_{ij}\left(\tilde{z}_i^{(s)} - \sum_{k<j} X_{ik}\tilde{\beta}_k^{(s,t)} - \sum_{k>j} X_{ik}\tilde{\beta}_k^{(s,t-1)}\right)\right| &= \frac{1}{n}\left|\sum_{i=1}^n \tilde{w}^{(s)} X_{ij}\left(\tilde{\beta}_0^{(s,0)} + \frac{Y_i - \tilde{p}^{(s)}}{\tilde{w}^{(s)}} - \tilde{\beta}_0^{(s,t)}\right)\right|\\ &= \frac{1}{n}\left|\sum_{i=1}^n \tilde{w}^{(s)} X_{ij}\left(\frac{Y_i - \tilde{p}^{(s)}}{\tilde{w}^{(s)}} - \frac{\bar{Y} - \tilde{p}^{(s)}}{\tilde{w}^{(s)}}\right)\right|\\ &= \frac{1}{n}\left|\sum_{i=1}^n X_{ij}\left(Y_i - \bar{Y}\right)\right|,\end{aligned}$$

$\tilde{\beta}_j^{(s,t)} = 0,\ j = 1, \ldots, p$ if and only if

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

Thus, λ_{max} as given in the theorem is the smallest value that can ensure that $\tilde{\beta}_j^{(s,t)} = 0$, $j = 1, \ldots, p$.

Given that $\tilde{\boldsymbol{\beta}}^{(0,0)} = \mathbf{0}$, and $\tilde{\boldsymbol{\beta}}^{(s-1,N_s)} = \tilde{\boldsymbol{\beta}}^{(s,0)}$, by induction, λ_{max} as given in the theorem is the smallest value that can ensure that $\forall s, t$, $\tilde{\beta}_j^{(s,t)} = 0$, $j = 1, \ldots, p$. □