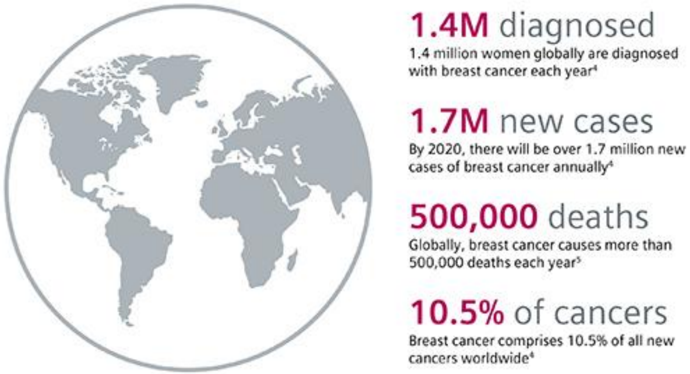
Breast Cancer Diagnosis

Wenhan Bao | Tianchuan Gao | Jialiang Hua | Qihang Wu | Paula Wu

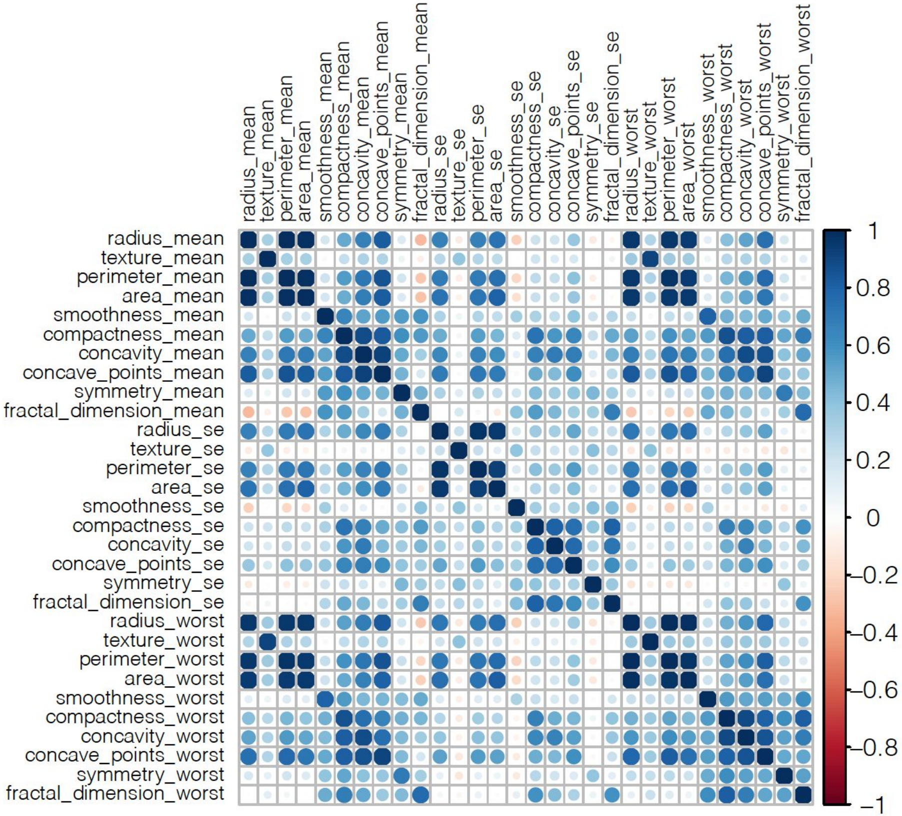
3/28/2022

Background

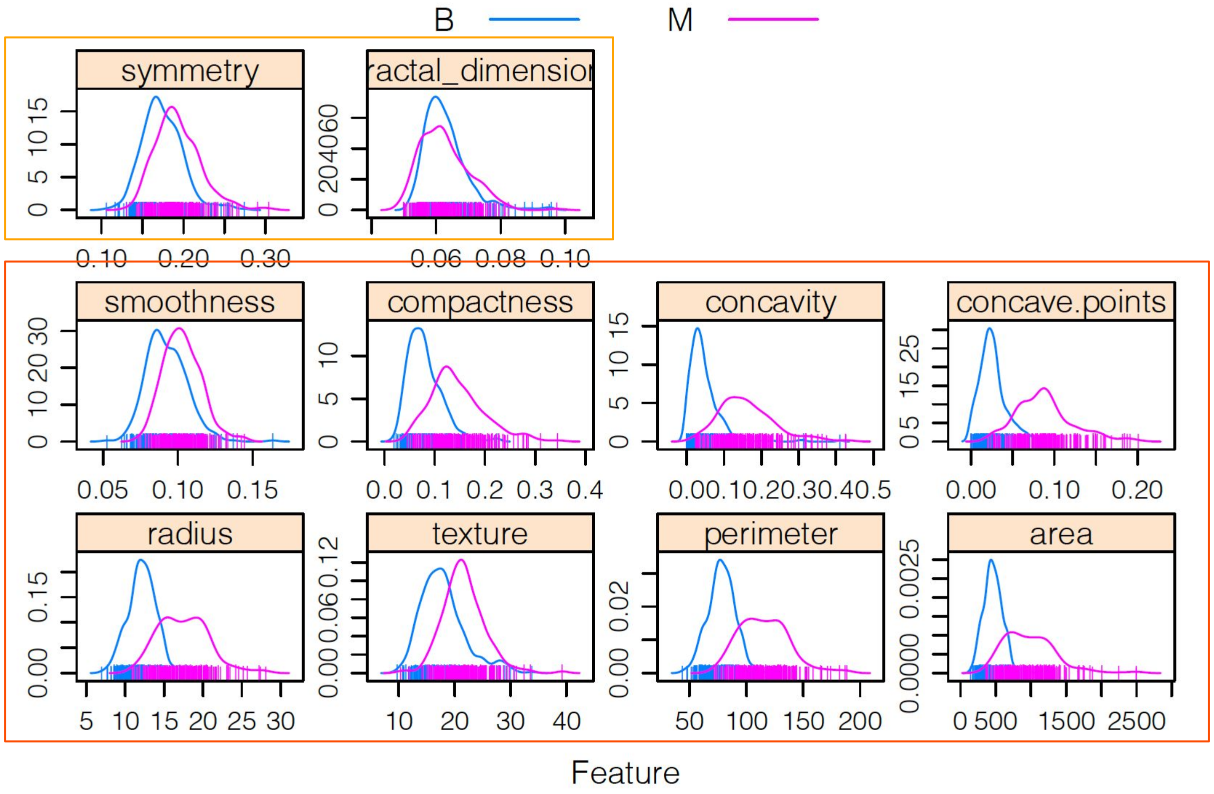
- Breast cancer: most common invasive cancer in women around the world
- Early accurate diagnosis can greatly improve prognosis
- Logistic model and logistic-LASSO model
- Estimation
  - Newton-Raphson algorithm
  - Pathwise coordinate-wise optimization algorithm



EDA



radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave_points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave_points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave_points_worst
symmetry_worst
fractal_dimension_worst
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1



B
M
symmetry
ractal_dimensior
0 5 10 15
0 204060
0.10 0.20 0.30
0.06 0.08 0.10
smoothness
compactness
concavity
concave.points
0 10 20 30
0 5 10
0 5 10 15
0 5 15 25
0.05 0.10 0.15
0.0 0.1 0.2 0.3 0.4
0.00.10.20.30.40.5
0.00 0.10 0.20
radius
texture
perimeter
area
0.00 0.15
0.000.060.12
0.00 0.02
0.0000 0.0025
5 10 15 20 25 30
10 20 30 40
50 100 150 200
0 500 1500 2500
Feature

Newton-Raphson Algorithm

The logistic model can be defined as: $log(\frac{\pi_i}{1-\pi_i}) = \mathbf{X}\boldsymbol{\beta} \rightarrow \pi_i = \frac{e^{\mathbf{X}\boldsymbol{\beta}}}{1+e^{\mathbf{X}\boldsymbol{\beta}}}$

The likelihood function : $L(\beta) = \prod_{i=0}^{n}(\frac{e^{\mathbf{X}\beta}}{1+e^{\mathbf{X}\beta}})^{y_i}(\frac{1}{1+e^{\mathbf{X}\beta}})^{1-y_i} \rightarrow$ Log likelihood function: $l(\beta) = \sum_{i=0}^{n}(y_i * X\beta - log(1+e^{X\beta}))$

Gradient: $\nabla l(\beta) = \begin{pmatrix} \sum_{i=1}^{n}(y_i - \pi_i) \\ \sum_{i=1}^{n} x_{i1} \times (y_i - \pi_i) \\ \cdot \\ \cdot \\ \cdot \\ \sum_{i=1}^{n} x_{in} \times (y_i - \pi_i) \end{pmatrix}$

Hessian Matrix:

$$\nabla^2 f(\beta) = -\sum_{i=1}^{n}\begin{pmatrix} 1 \\ x_i \end{pmatrix}\begin{pmatrix} 1 & x_i \end{pmatrix}\pi_i(1-\pi_i)$$

$$= -\begin{pmatrix} \sum \pi_i(1-\pi_i) & \sum x_i\pi_i(1-\pi_i) \\ \sum x_i\pi_i(1-\pi_i) & \sum x_i^2\pi_i(1-\pi_i) \end{pmatrix}$$

With these parameters, we can apply the Newton-Raphson algorithm to calculate a set of beta using this equation:

$$\boldsymbol{\beta}_{\boldsymbol{i+1}} = \boldsymbol{\beta}_{\boldsymbol{i}} - \left[\nabla^2 l(\boldsymbol{\beta}_{\boldsymbol{i}})\right]^{-1} \nabla l(\boldsymbol{\beta}_{\boldsymbol{i}})$$

Step-halving and re-direction modification:
Ensure the likelihood is increasing and the ascent direction at $\boldsymbol{\beta}_i$

Newton-Raphson Result

- Test for the first 10 features.
- Compare to glm results, outcomes are very close.
- Also test for all 30 features, result is not ideal due to computation burden when calculating the inverse of the Hessian matrix.

	Coefficients
Intercept	0.487
Radius Mean	-7.221
Texture Mean	1.654
Perimeter Mean	-1.737
Area Mean	14.005
Smoothness Mean	1.075
Compactness Mean	-0.077
Concavity Mean	0.675
Concave Point Mean	2.593
Symmetry Mean	0.446
Fractal Dimension Mean	-0.482

```
       (Intercept)          radius_mean         texture_mean       perimeter_mean            area_mean      smoothness_mean 
        0.48701675          -7.22185053           1.65475615          -1.73763027          14.00484560           1.07495329 
  compactness_mean       concavity_mean  concave_points_mean        symmetry_mean fractal_dimension_mean 
       -0.07723455           0.67512313           2.59287426           0.44625631          -0.48248420 
```

Coordinatewise descent (CD) Lasso

- **High-dimensional covariates; Standardize**
- **Soft threshold function:** $S\left(\hat{\beta}, \gamma\right) = \text{sign}\left(\hat{\beta}\right)\left(\left|\hat{\beta}\right| - \gamma\right)_+$
- **CD algorithm: update** β_j **repeatedly for j = 1, …, p, 1, …, p**
- For different weights w_i : $\tilde{\beta}_j(\gamma) = \dfrac{S\left(\sum_i^n w_i x_{i,j}\left(Z_i - \tilde{Z}_i^{(-j)}\right), \gamma\right)}{\sum_i^n w_i x_{i,j}^2}$
- Working weights $w_i > 0$: $\tilde{\pi}_i(1 - \tilde{\pi}_i)$
- Working response Z_i : $X_i\tilde{\beta} + \dfrac{y_i - \tilde{\pi}_i}{\pi_i(1 - \pi_i)}$

Taylor expansion around current estimates:

$$l(\beta) = -\frac{1}{2n}\sum_{i=1}^{n} w_i(Z_i - X_i\beta)^2$$

- **Object Function:** $\boxed{\min_{(\beta)} L(\beta, \lambda) = \left\{-l(\beta) + \lambda\sum_{j=0}^{p}|\beta_j|\right\}}$

```r
# Soft Threshold function
sf <- function(beta, lambda) {
  beta <- ifelse(lambda < abs(beta),
                 ifelse(beta > 0, beta - lambda, beta + lambda), 0)
  return(beta)
}


# Coordinate-wise LASSO with fixed lambda
cd_lasso <- function(dat, y, betavec, lambda, maxier = 2000, tol = 1e-10) {
  x <- dat # n * (p + 1) matrix, standardized
  i <- 0
  objfun <- 0
  prev_objfun <- -Inf # ensure iterations
  res <- c(0, objfun, betavec)

  while (i < maxier && abs(objfun - prev_objfun) > tol) {
    i <- i + 1
    prev_objfun <- objfun

    for (j in 1:length(betavec)) {
      u <- x %*% betavec
      pi <- exp(u) / (1 + exp(u))
      # working weights
      w <- pi * (1 - pi)
      w <- ifelse(w < 1e-5, 1e-5, w)
      # working response
      z <- x %*% betavec + (y - pi) / w
      z_dej <- x[, -j] %*% betavec[-j]
      betavec[j] <-
        sf(sum(w * x[, j] * (z - z_dej)), lambda) / (sum(w * x[, j]^2))
    }
    objfun <-
      sum(w * (z - x %*% betavec)^2) / (2 * dim(x)[1]) + lambda * sum(abs(betavec))
    if (is.na(objfun)) {break}

    res <- rbind(res, c(i, objfun, betavec))
  }

  return(res)
}
```

LASSO - Pathwise Coordinate-wise optimization

1. Select a λmax for all the estimate β = 0
2. Compute the solution with a sequence of descending λ from maximum
3. Initialize coordinate descent algorithms by the calculated estimate β from previous λ
4. Find the optimal solution by comparing the objective function:

$$\min_{(\beta)} L(\beta, \lambda) = \left\{ -l(\beta) + \lambda \sum_{j=0}^{p} |\beta_j| \right\}$$

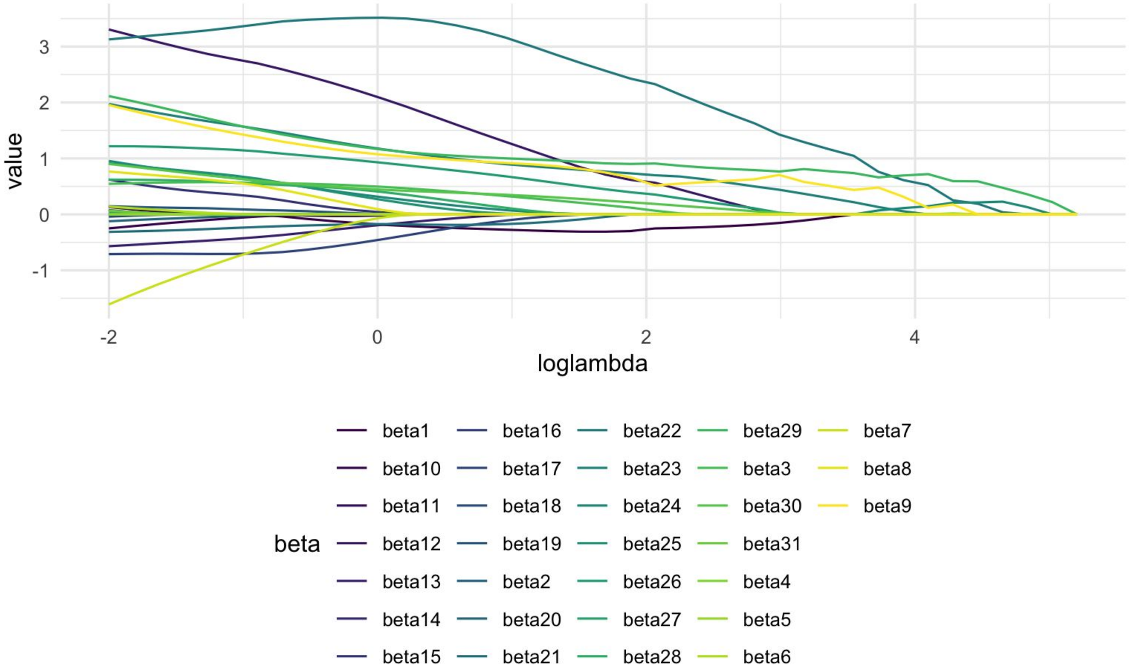
```
> max(t(bc_scale2_x) %*% bc_scale2_y)
[1] 218.1238
```

```
pathwise_cd_lasso <- function(dat, y, betavec, lambda) {

  sort(lambda, decreasing = TRUE)
  pathwise_res <- NA

  for (j in 1:length(lambda)) {
    cd_results <- cd_lasso(dat, y, betavec, lambda[j])

    # warm start
    min_objfun_ind <- which.min(cd_results[-1, 2]) + 1
    betavec <- cd_results[min_objfun_ind, -c(1, 2)]
    objfun <- cd_results[min_objfun_ind, 2]

    pathwise_res <- rbind(pathwise_res,
                          c(j, lambda = lambda[j], obj = objfun,
                            beta = betavec))

  }

  return(pathwise_res[-1, ])
}
```

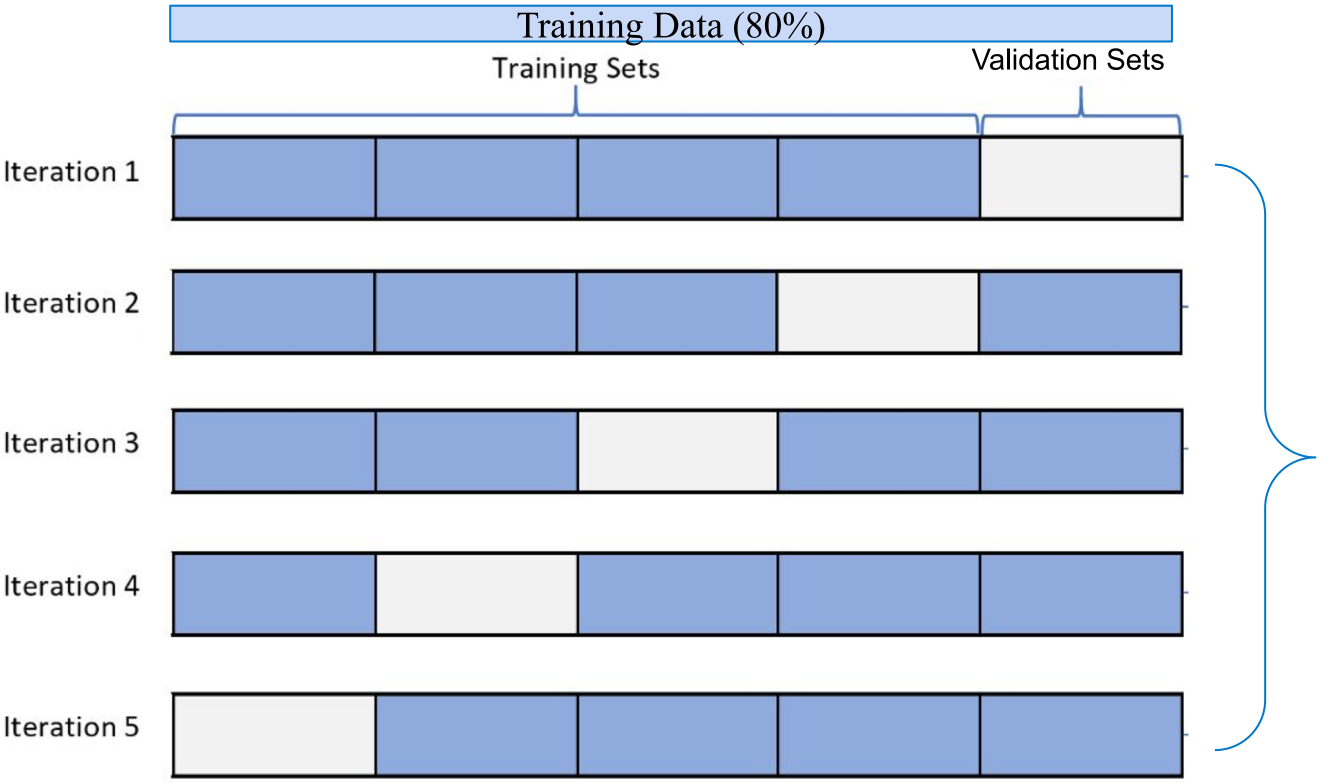
LASSO - Pathwise Coordinate-wise optimization

- Solution for each coefficients β under different descending λ

- Roughly, λ we select is 0.135 but need further cross-validation



Cross Validation - Overview



Testing Data (20%)

Best λ Model Evaluation

Obtain:
1. Mean/sd AUC
2. Residual Sum of Squares

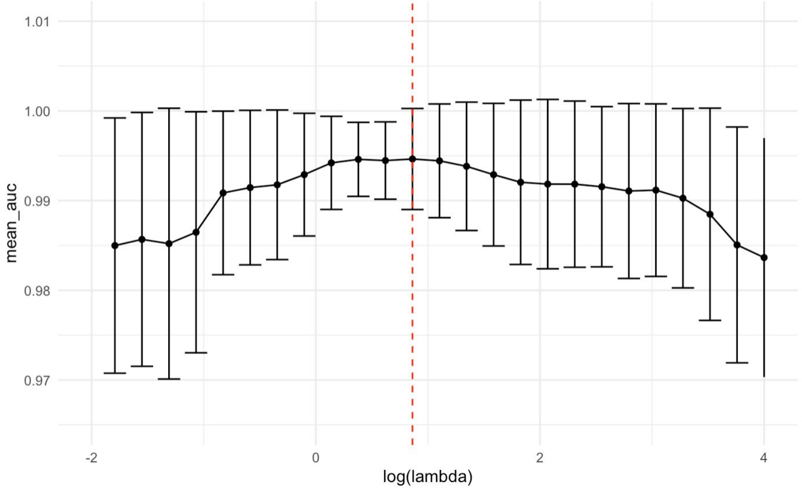
[2]

Cross Validation - Steps

1. Select 30 equally spaced λ from range $e^{(4, -3)}$
   a. Upper bound: arbitrary, a λ value that is large but not large enough to force all β's become 0.
2. For each λ, run the 5-fold CV. For each fold:
   a. Run Coordinate-wise Lasso on training sets
   b. Extract current "best" beta (i.e. the last row) from the results
   c. Make prediction and calculate AUC, RSS against validation sets
3. Calculate mean and std-dev of AUC and mean RSS for each λ

Results & Discussion - Optimal vs. Full Models

- Optimal λ: the λ that maximize the mean AUC
  - ~ 2.3681
- Test against testing data:
  - Optimal Model: λ_optimal = 2.3681
  - Full Model: λ = 0
- Compare the predicted outcome vs. actual outcome → calculate AUC
- Full model: AUC = 1
- Optimal model: AUC = 0.9989



Limitations

- Our Newton-Raphson method still use the inverse of Hessian matrix, which is not accurate when p is large
  - Try alternatives that replace Hessian matrix

- During the cross-validation process, we select a wide range of lambda with only sparse data points.
  - Try a denser model with narrower range for more precise predict of λ

Questions?

Reference

[1] Data: breast_cancer.csv

[2] CV illustration:

https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-e54df2fc179b

Thank you!