Task 4

Task 4: Use 5-fold cross-validation to select the best λ. Compare the prediction performance between the "optimal" model and "full" model.

5-fold CV

We write an **R** function `cv.logit.lasso` to conduct 5-fold cross-validation to select the best λ.

```
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)
}
```

Compare the results of cross-validation using `glmnet` and using our algorithm.

1. Our function `cv.logit.lasso`:

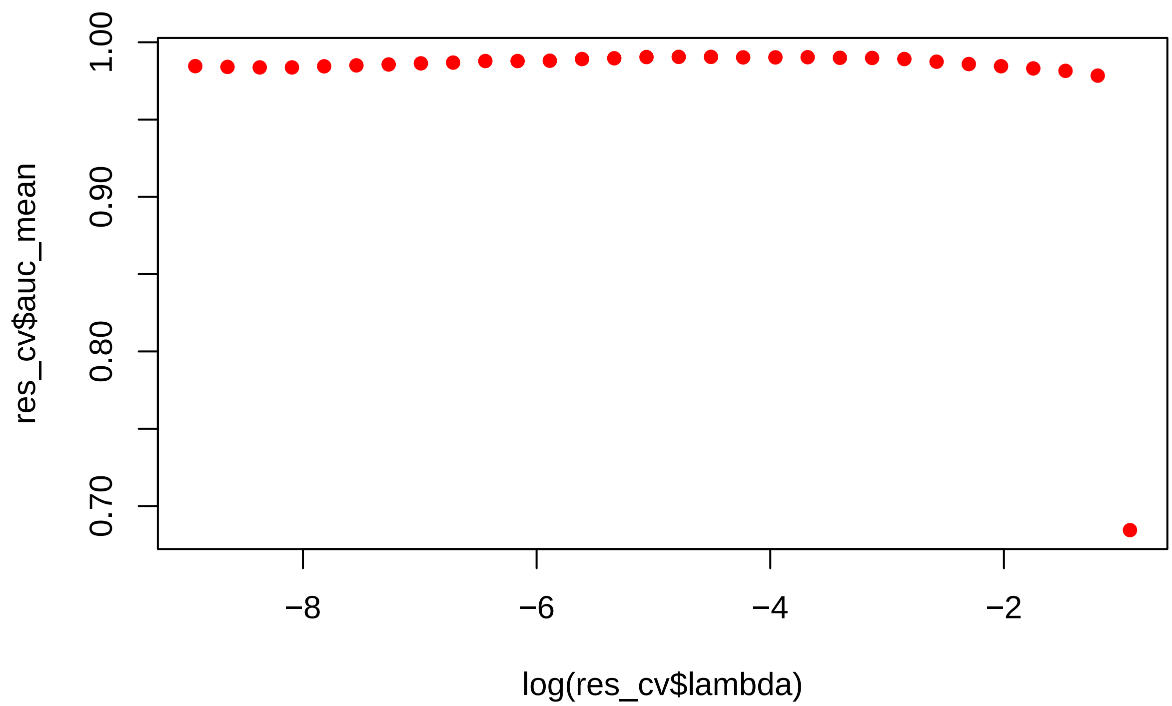
```
lambda_max <- max(abs(t(x) %*% y)) / length(y)
lambdas <- exp(seq(log(lambda_max), log(lambda_max) - 8, length = 30))
set.seed(1)
res_cv = cv.logit.lasso(x, y, nfolds = 5, lambda = lambdas)
as.matrix(res_cv %>% arrange(-lambda))
```

```
##              lambda  auc_mean auc_ranking
##  [1,] 0.3979882278 0.6844328          30
##  [2,] 0.3020402714 0.9784198          29
##  [3,] 0.2292236784 0.9814917          28
##  [4,] 0.1739618843 0.9830838          27
##  [5,] 0.1320227361 0.9845052          22
##  [6,] 0.1001943782 0.9859027          18
##  [7,] 0.0760392772 0.9874342          15
##  [8,] 0.0577075459 0.9891438          10
##  [9,] 0.0437952724 0.9898392           8
## [10,] 0.0332370031 0.9899387           7
## [11,] 0.0252241467 0.9903360           4
## [12,] 0.0191430489 0.9902353           5
## [13,] 0.0145279968 0.9902346           6
## [14,] 0.0110255525 0.9905325           1
## [15,] 0.0083674858 0.9905317           2
## [16,] 0.0063502323 0.9904323           3
## [17,] 0.0048193031 0.9896377           9
## [18,] 0.0036574539 0.9891413          11
## [19,] 0.0027757062 0.9880447          12
## [20,] 0.0021065323 0.9878500          14
## [21,] 0.0015986844 0.9878512          13
## [22,] 0.0012132697 0.9868503          16
## [23,] 0.0009207718 0.9863493          17
## [24,] 0.0006987899 0.9856446          19
## [25,] 0.0005303240 0.9850432          20
## [26,] 0.0004024722 0.9844403          23
## [27,] 0.0003054432 0.9837394          25
## [28,] 0.0002318062 0.9837389          26
## [29,] 0.0001759218 0.9840401          24
## [30,] 0.0001335102 0.9845400          21
```

```r
# best lambda
best_lambda <- max(res_cv$lambda[res_cv$auc_ranking == 1])
best_lambda
```

```
## [1] 0.01102555
```

```r
plot(log(res_cv$lambda), res_cv$auc_mean, pch = 16, col = "red")
```




```
# coefficients of the best model
res_coef <- LogisticLASSO(dat = Training, start = rep(0, ncol(Training)),
                          lambda = lambdas) %>% as.data.frame
res_coef[res_coef$lambda == best_lambda, -1]
```

```
##    (Intercept) radius_mean texture_mean perimeter_mean area_mean
## 14  -0.6429962           0    0.3643606              0         0
##    smoothness_mean compactness_mean concavity_mean concave.points_mean
## 14               0                0              0            0.486222
##    symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 14             0                      0 0.6235709          0            0
##    area_se smoothness_se compactness_se concavity_se concave.points_se
## 14       0             0              0            0                 0
##    symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 14           0                    0     2.490748     0.4262261               0
##    area_worst smoothness_worst compactness_worst concavity_worst
## 14          0        0.4563645                 0      0.09595825
##    concave.points_worst symmetry_worst fractal_dimension_worst
## 14            0.9998579      0.3463273                       0
```

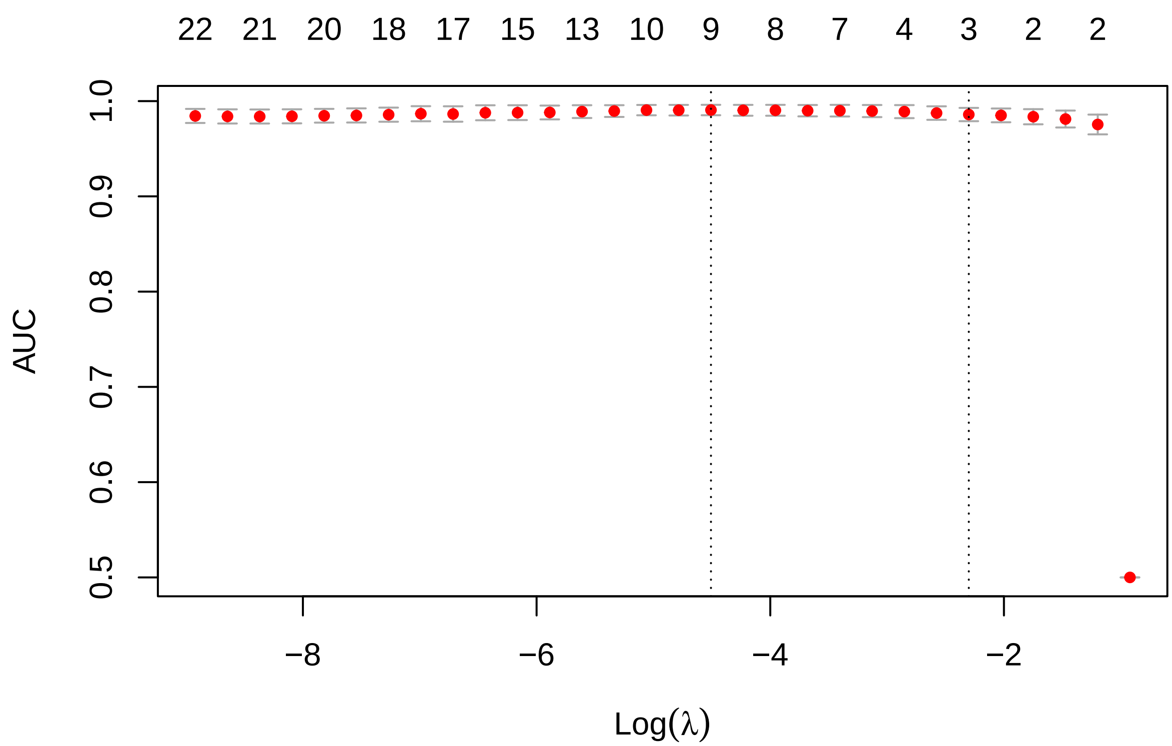
2. glmnet from **R** package caret

```
set.seed(1)
fit.logit.lasso <- cv.glmnet(x, y,
                             nfolds = 5, alpha = 1,
```

```
                              lambda = lambdas,
                              family = "binomial", type.measure = "auc")
# best lambda
fit.logit.lasso$lambda.min
```

```
## [1] 0.01102555
```

```
plot(fit.logit.lasso)
```




```
# coefficients of the best model
coef(fit.logit.lasso, fit.logit.lasso$lambda.min)
```

```
## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)             -0.62737091
## radius_mean              .
## texture_mean             0.35254009
## perimeter_mean           .
## area_mean                .
## smoothness_mean          .
## compactness_mean         .
## concavity_mean           .
## concave.points_mean      0.43620649
## symmetry_mean            .
```

```
## fractal_dimension_mean    .
## radius_se                 0.72496571
## texture_se                .
## perimeter_se              .
## area_se                   .
## smoothness_se             .
## compactness_se            .
## concavity_se              .
## concave.points_se         .
## symmetry_se               .
## fractal_dimension_se      .
## radius_worst              2.50424036
## texture_worst             0.45160840
## perimeter_worst           .
## area_worst                .
## smoothness_worst          0.46310471
## compactness_worst         .
## concavity_worst           0.07948187
## concave.points_worst      1.07060218
## symmetry_worst            0.35124064
## fractal_dimension_worst   .
```

The results are slightly different (mean AUC values).

```r
tibble(
  lambda = lambdas,
  ours_AUC = res_cv %>% arrange(-lambda) %>% .$auc_mean,
  cv.glmnet_AUC = fit.logit.lasso$cvm
) %>%
  knitr::kable()
```

lambda	ours_AUC	cv.glmnet_AUC
0.3979882	0.6844328	0.5000000
0.3020403	0.9784198	0.9754654
0.2292237	0.9814917	0.9811797
0.1739619	0.9830838	0.9835991
0.1320227	0.9845052	0.9850204
0.1001944	0.9859027	0.9859165
0.0760393	0.9874342	0.9874491
0.0577075	0.9891438	0.9889595
0.0437953	0.9898392	0.9895565
0.0332370	0.9899387	0.9899549
0.0252241	0.9903360	0.9899533
0.0191430	0.9902353	0.9903517
0.0145280	0.9902346	0.9903518
0.0110256	0.9905325	0.9906491
0.0083675	0.9905317	0.9904501
0.0063502	0.9904323	0.9905479
0.0048193	0.9896377	0.9895550
0.0036575	0.9891413	0.9889605
0.0027757	0.9880447	0.9880654
0.0021065	0.9878500	0.9878701

lambda	ours_AUC	cv.glmnet_AUC
0.0015987	0.9878512	0.9877668
0.0012133	0.9868503	0.9864656
0.0009208	0.9863493	0.9867637
0.0006988	0.9856446	0.9857520
0.0005303	0.9850432	0.9849479
0.0004025	0.9844403	0.9846392
0.0003054	0.9837394	0.9840368
0.0002318	0.9837389	0.9838370
0.0001759	0.9840401	0.9839377
0.0001335	0.9845400	0.9844378

The best λ's are the same, and the coefficients are very similar.

```r
# our best lambda
best_lambda
```

```
## [1] 0.01102555
```

```r
# cv.glmnet's best lambda
fit.logit.lasso$lambda.min
```

```
## [1] 0.01102555
```

```r
tibble(
  predictor = c("(Intercept)", names(Training)[-1]),
  ours_coef = res_coef[res_coef$lambda == best_lambda, -1] %>% as.vector %>% as.numeric,
  cv.glmnet_coef = coef(fit.logit.lasso, fit.logit.lasso$lambda.min) %>% as.vector
) %>%
  knitr::kable()
```

predictor	ours_coef	cv.glmnet_coef
(Intercept)	-0.6429962	-0.6273709
radius_mean	0.0000000	0.0000000
texture_mean	0.3643606	0.3525401
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	0.4862220	0.4362065
symmetry_mean	0.0000000	0.0000000
fractal_dimension_mean	0.0000000	0.0000000
radius_se	0.6235709	0.7249657
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

predictor	ours_coef	cv.glmnet_coef
concavity_se	0.0000000	0.0000000
concave.points_se	0.0000000	0.0000000
symmetry_se	0.0000000	0.0000000
fractal_dimension_se	0.0000000	0.0000000
radius_worst	2.4907478	2.5042404
texture_worst	0.4262261	0.4516084
perimeter_worst	0.0000000	0.0000000
area_worst	0.0000000	0.0000000
smoothness_worst	0.4563645	0.4631047
compactness_worst	0.0000000	0.0000000
concavity_worst	0.0959582	0.0794819
concave.points_worst	0.9998579	1.0706022
symmetry_worst	0.3463273	0.3512406
fractal_dimension_worst	0.0000000	0.0000000

Prediction performance comparison

We probably need resampling methods (conducted in training data) to select the best model. Is the resampling methods in task 2 correct?

Below is the prediction performance on the test data. *(I suppose this should not be used for model comparison)*

```
# test data
X_test <- cbind(rep(1, nrow(Test)), model.matrix(diagnosis ~ ., Test)[, -1])
y_test <- Test$diagnosis

# logistic model
res_logit <- NewtonRaphson(dat = Training, func = logisticstuff, start = rep(0, ncol(Training)))
```

```
## Warning in NewtonRaphson(dat = Training, func = logisticstuff, start = rep(0, :
## Complete separation occurs. Algorithm does not converge.
```

```
betavec_logit <- res_logit[nrow(res_logit), 3:ncol(res_logit)]
u <- X_test %*% betavec_logit
phat <- sigmoid(u)[, 1]
roc.logit <- roc(response = y_test, predictor = phat)

# logistic LASSO model
betavec_logit.lasso <- res_coef[res_coef$lambda == best_lambda, -1] %>% as.vector %>% as.numeric
u <- X_test %*% betavec_logit.lasso
phat <- sigmoid(u)[, 1]
roc.logitlasso <- roc(response = y_test, predictor = phat)

# logistic LASSO model (cv.glmnet)
betavec_logit.lasso.glm <- coef(fit.logit.lasso, fit.logit.lasso$lambda.min) %>% as.vector
u <- X_test %*% betavec_logit.lasso.glm
phat <- sigmoid(u)[, 1]
roc.logitlasso.glm <- roc(response = y_test, predictor = phat)

# draw rocs
auc <- c(roc.logit$auc[1], roc.logitlasso$auc[1], roc.logitlasso.glm$auc[1])
```

```r
plot(roc.logit, legacy.axes = TRUE)
plot(roc.logitlasso, col = 2, add = TRUE)
plot(roc.logitlasso.glm, col = 3, add = TRUE)
modelNames <- c("logistic", "logistic LASSO", "logistic LASSO (cv.glmnet)")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc, 3)),
col = 1:3, lwd = 2)
```