Statistical Methods for High Dimensional Biology

STAT/BIOF/GSAT 540

Lecture 19 – Regularization

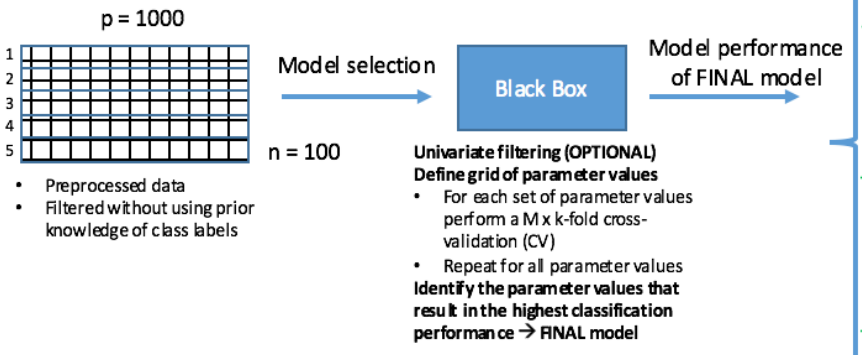
Amrit Singh

March 20 2017

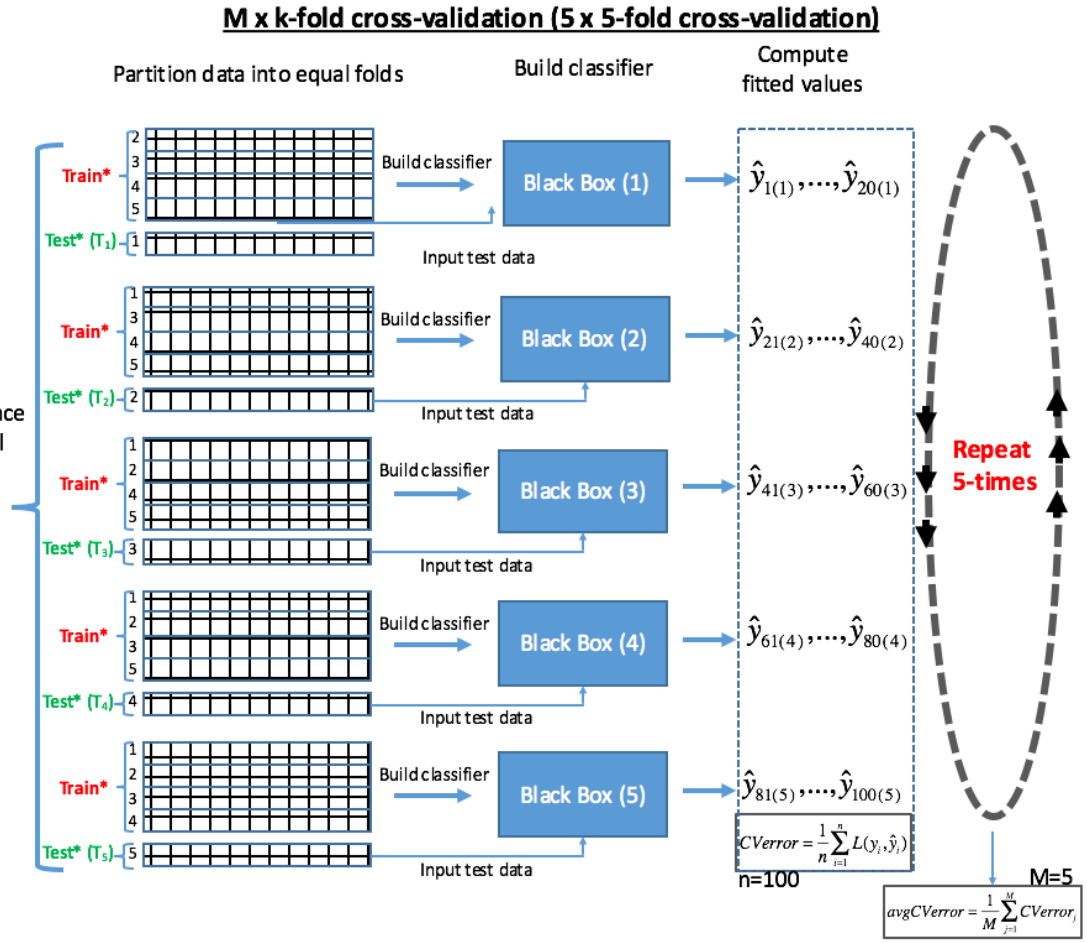
Based on slides by Drs. Cohen Freue, Mostafavi & Pavlidis

Review – cross-validation

Model selection and Model assessment



M = number of iterations (repetitions)
k = number of folds (1-n), where n is the number of samples
*the center and scale values of the training dataset are used to standardize both the training and test data



Outline

Shrinkage methods: fit model containing p predictors then **regularize** the coefficient estimates (**shrinkage** estimates towards zero)

- Ridge regression
- Lasso (least absolute shrinkage and selection operator)

Application:

- Compare classification models

Feature selection in linear regression

Best subset selection

- Select best model from all possible models (each model can have a different combination of predictors): 2^p possible models

Forward Stepwise Selection

- Starting with the null model (Intercept), sequentially add the predictor that improves the model fit (e.g. reduces RSS, improve R^2): $p(p+1)/2 + 1$ possible models

Backward Stepwise Selection

- Starting with the full model (containing all predictors), sequentially remove the least useful predictor (e.g. reduces RSS, improve R^2): $p(p+1)/2 + 1$ possible models

Select best model based on C_p (Mallow's C_p), AIC, BIC, Adjusted R^2 or cross-validation

James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning*. **103,** (Springer New York, 2013).

Choosing the optimal model

<u>Adjust training error</u>

C_p (Mallow's C_p), AIC, BIC (best model: model with the lowest value)

- Apply a penalty to the training error to adjust for overfitting model to training data
- Make theoretical assumption about the distribution of the errors

Adjusted R^2 (best model: model with the highest value)

- R^2 is modified to account for the number of variables in the model

<u>Estimate test error</u>

Cross-validation

- Fewer assumptions about the true underlying model
- Can be used if it is hard to estimate the number of degrees of freedom or estimate error variance

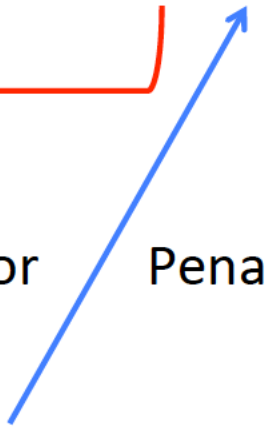
James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning*. **103,** (Springer New York, 2013).

Regularized regression

$$\underset{\beta}{\operatorname{argmin}} \; (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta})^T (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta})$$

Squared error

Penalization function



Penalization (tuning) parameter

Regularized regression: ridge

$$\underset{\beta}{\operatorname{argmin}} \; (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta})^T(\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta})$$

Squared error

Penalization function

$$P(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^{p} \beta_i^2$$

Ridge regression: L2 norm ("squared") penalty function

$\lambda \geq 0$ is the penalization parameter that we can "tune"

- When $\lambda = 0$ we get the linear regression estimate
- When $\lambda = \infty$ what happens?
- For λ in between 0 and infinity we are balancing error term vs penalty

Regularized regression: ridge

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta})^T (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta}) + \boldsymbol{\lambda} \left\| \boldsymbol{\beta} \right\|_2^2$$

Squared error

Penalization function

$$\boldsymbol{\beta}^{ridge} = (X^T X + \boldsymbol{\lambda} I)^{-1} X^T \boldsymbol{y}$$

Regularized regression: ridge

$$\underset{\beta}{\operatorname{argmin}} \; (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta})^T (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta})$$

Squared error Penalization function

$$P(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^{p} \beta_i^2$$

Ridge regression: L2 norm ("squared") penalty function

Important details:

- How do you choose λ ? Need nested CV to tune parameter and estimate test error.
- Requires standardization of the covariates (columns of X) :
  - the covariates need to be on the same "scale"
- Since columns standardized no intercept required

Regularized regression: lasso

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \ (y\text{-}X\beta)^T (y\text{-}X\beta) + \lambda \|\beta\|_1$$

$$\|\beta\|_1 = \sum_{i=1}^{p} |\beta_i| \longrightarrow \text{L1 norm penalty}$$

- Proposed by Tibshirani (1996)
- Lasso is an acronym: Least Absolute Selection and Shrinkage Operator
- The only difference between ridge and lasso regularization is that ridge uses L2 norm vs lasso uses L1 norm. But turns out that the solutions to lasso and ridge behaves very differently.

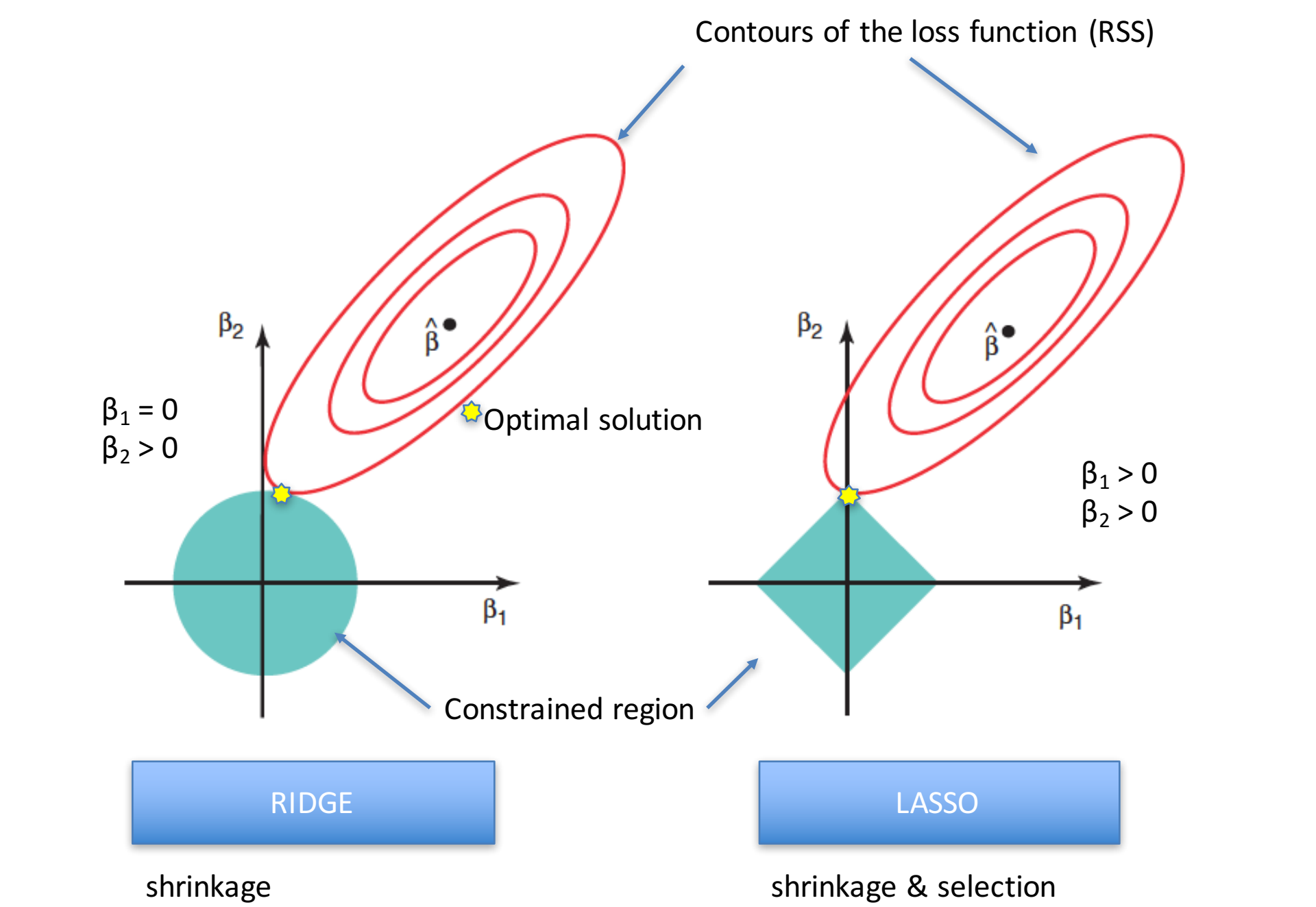
$$\hat{\boldsymbol{\beta}}_{lasso} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \ (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta})^T (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta}) + \lambda \left\| \boldsymbol{\beta} \right\|_1$$

As before:

- The tuning parameter $\lambda \geq 0$ controls the strength of the penalty. What happens at the 0 and infinity?

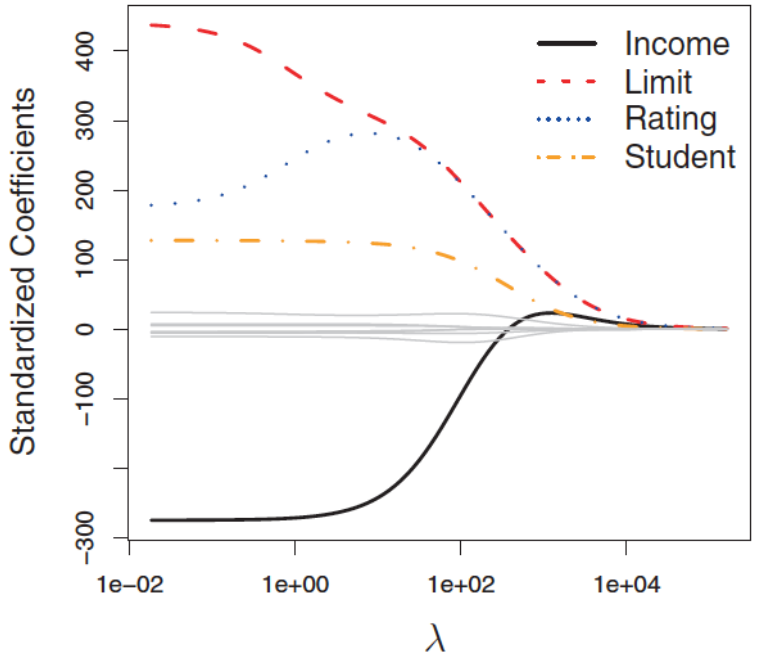
But now:

- For $\lambda \geq 0$ between the two extreme setting, we are balancing two ideas: minimizing the squared error, and penalizing the magnitude of the coefficient. But the L1 penalty causes some coefficients to bet set to **zero exactly**.
- Because some coefficients are set to zero exactly, lasso performs variable selection.
- No closed form solution: requires numerical optimization.
  - LARS algorithm
  - ADMM algorithm **



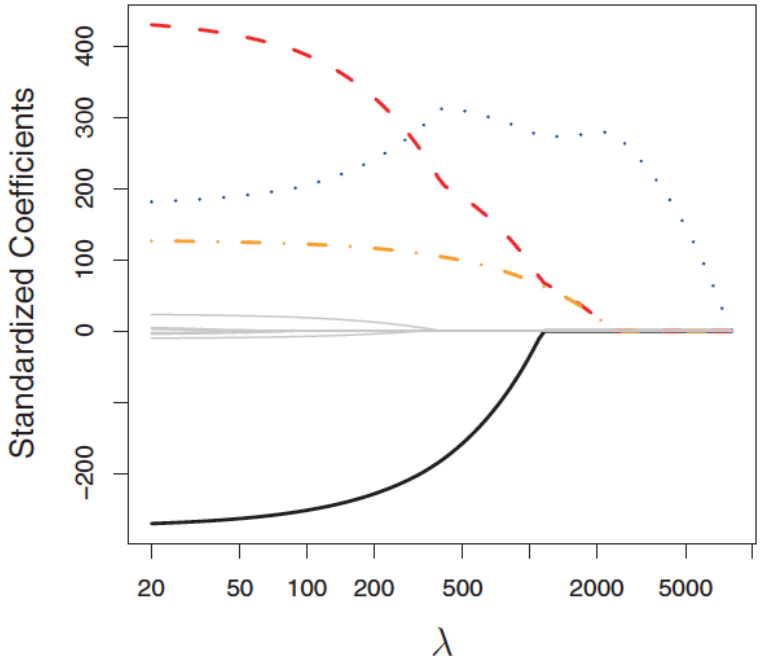
James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning*. **103,** (Springer New York, 2013).

RIDGE



λ=0 → Least squares solution

LASSO



λ=large → model includes no predictors

Ridge

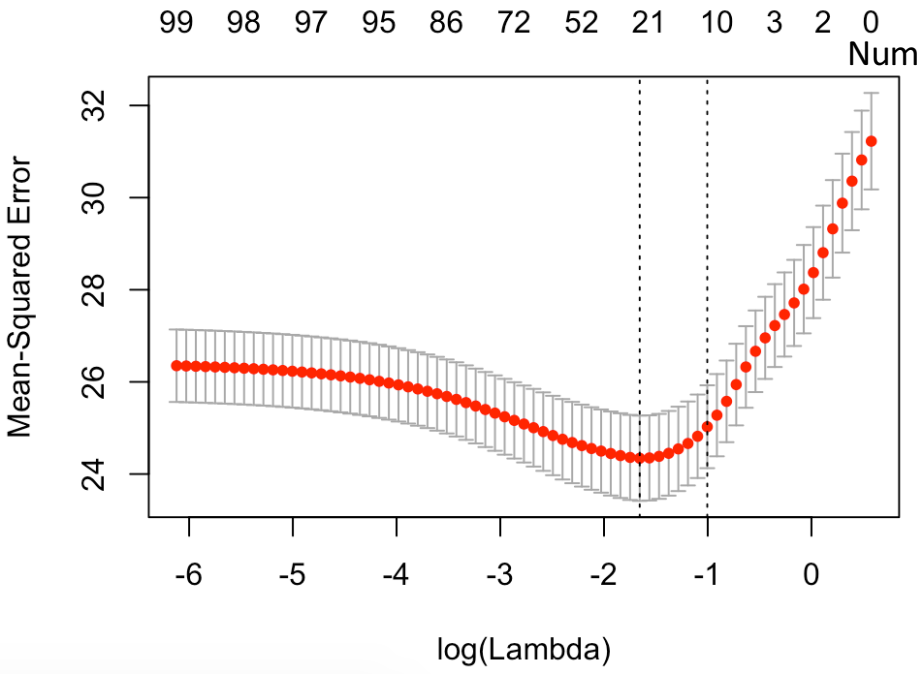
- All coefficients are shrunk towards zero but almost never equal zero (reduces variance)

Lasso:

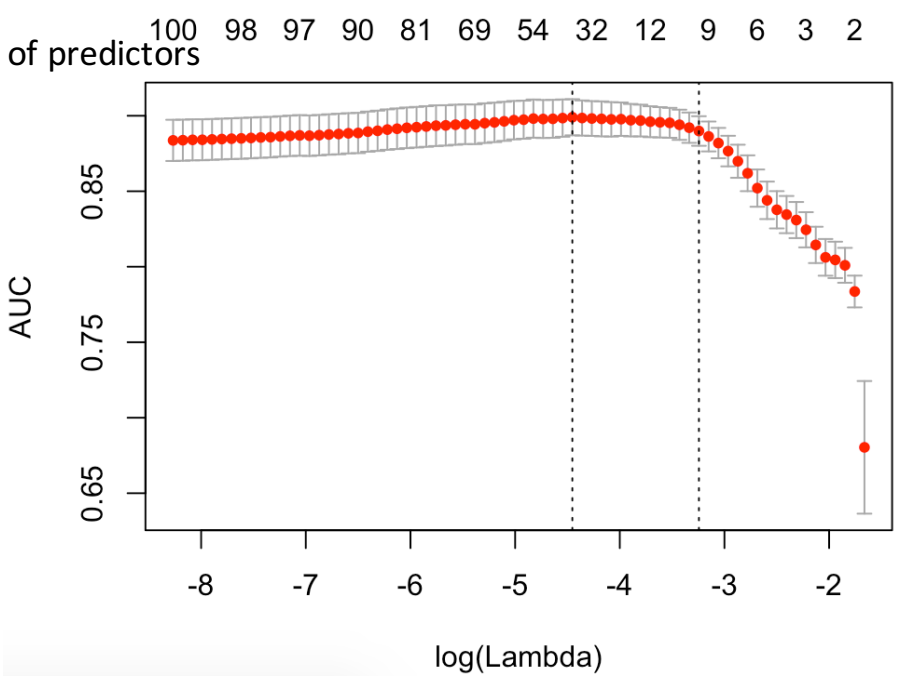
- All coefficients are shrunk towards zero and many will equal exactly zero (these predictors have no contribution in the prediction) – reduces variance with a little increase in bias (since the true coefficient estimates may not exactly equal zero)

How to choose the optimal value of lambda?

response (y) is continuous



response (y) is binary



?glmnet::cv.glmnet()

Select the lambda value that corresponds to the model with the lowest error rate, or highest classification performance.

Limitations of Ridge and Lasso

- In general, ridge does not give a sparse solution (does not perform variable selection method)
- If $p > n$, lasso can select at most n variables out of p candidates (Efron et al., 2004)

- If there is a group of highly correlated variables, lasso tends to select only one covariate from the group.

- Lasso will perform better if only a small set of predictors is associated with the response

- Neither methods is better than the other; use cross-validation to determine which approach is better on a given dataset

The elastic net: combined L2 and L1 penalty

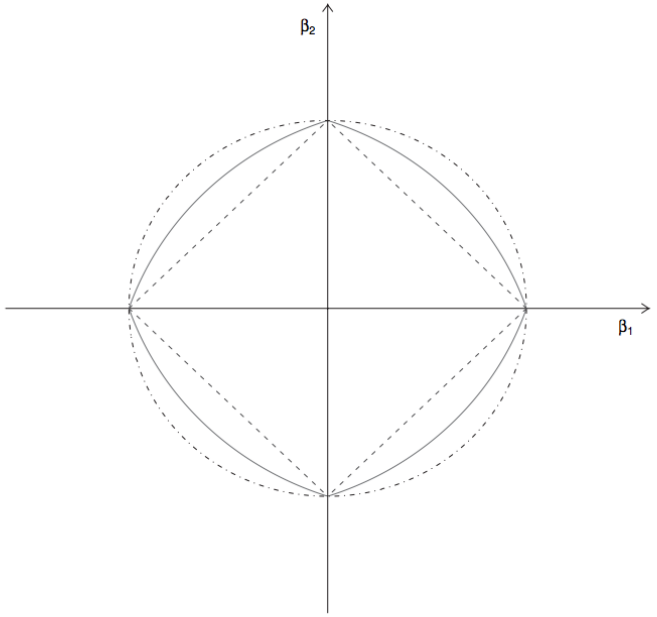


Fig. 1. Two-dimensional contour plots (level 1) (·-·-·-·, shape of the ridge penalty; -------, contour of the lasso penalty; ———, contour of the elastic net penalty with $\alpha = 0.5$): we see that singularities at the vertices and the edges are strictly convex; the strength of convexity varies with α

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta})^T (\boldsymbol{y}\text{-}\mathrm{X}\boldsymbol{\beta}) + \lambda((1-\alpha)\|\boldsymbol{\beta}\|_1 + \alpha\|\boldsymbol{\beta}\|_2^2)$$

- Convex combination of the L1 and L2 penalty
- Addresses the shortcoming of the lasso in the presence of correlated features.
- But now we have to set 2 parameters.

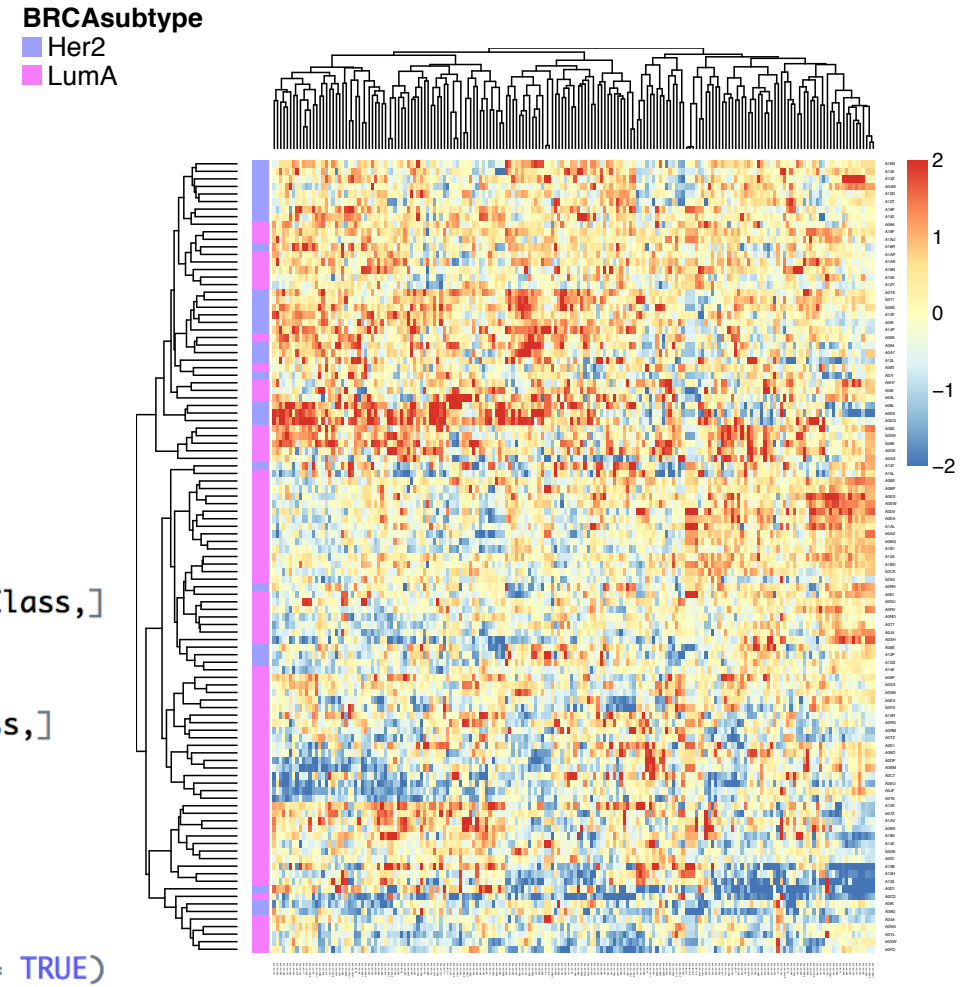
J. R. Statist. Soc. B (2005) 67, Part 2, pp. 301–320

Choosing λ

- The function `cv.glmnet` runs glmnet to get the λ values for which an additional coefficient is added to the model.
- Given this sequence, cv.glmnet does an n-fold cross-validation (default n = 10) and estimates the prediction error.
- The average error and standard deviation over the folds can be plotted to choose the optimal λ. Note that cv.glmnet does NOT search for values for alpha.
- lambda.min gives the λ that minimizes the error. lambda.1se is the largest value of lambda with an error within 1 standard error from the minimum (i.e., a less complex model at a low cost).

Identify a miRNA biomarker panel that can discriminate breast cancer subtypes

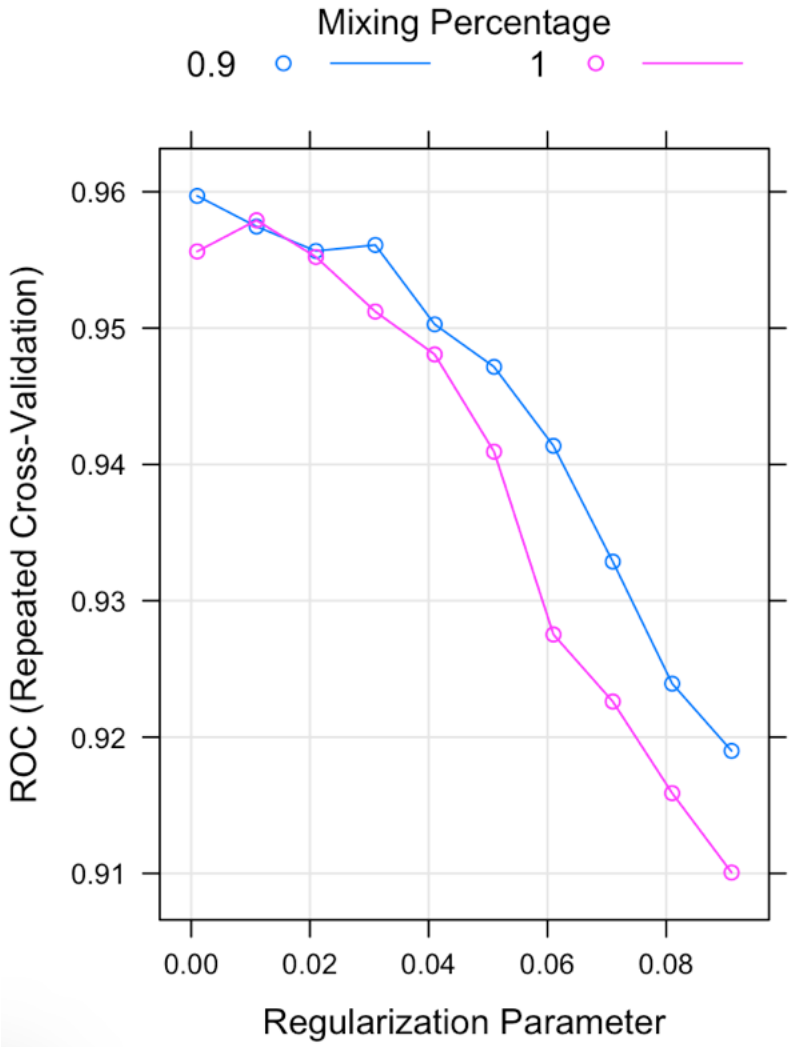
150 samples x 184 miRNAs
30 Her2
75 LumA




```
library(mixOmics) # to obtain Breast Cancer data
library(ggbiplot) # plot PCA plot
library(caret)  # classifier functions

## Import data
rmClass <- "Basal"
## Train and Test data
y.train <- breast.TCGA$data.train$subtype
X.train <- breast.TCGA$data.train$mirna[y.train != rmClass,]
y.train <- droplevels(y.train[y.train != rmClass])
y.test <- breast.TCGA$data.test$subtype
X.test <- breast.TCGA$data.test$mirna[y.test != rmClass,]
y.test <- droplevels(y.test[y.test != rmClass])
levels(y.train) == levels(y.test)

## heatmap of miRNA dataset
dim(X.train); table(y.train);
X.trainScaled <- scale(X.train, center = TRUE, scale = TRUE)
X.trainScaled[X.trainScaled < -2] <- -2
X.trainScaled[X.trainScaled > 2] <- 2
colors <- RColorBrewer::brewer.pal(3, "Spectral")
NMF::aheatmap(X.trainScaled, annRow = list(BRCAsubtype = y.train),
  filename = "~/Documents/Courses/STAT540/lect19_regularization/heatmap.pdf")
```

1) Determine biomarker panel using elastic net (Enet)

```r
## set parameters for repeatedCV
M = 2
k = 2

## Set options
set.seed(11)
enetControl <- trainControl(method = "repeatedcv",
  number = k, repeats = M,
  classProbs = TRUE,
  summaryFunction = twoClassSummary)
enetGrid = expand.grid(alpha = c(0.9, 1),
  lambda = seq(0.001, 0.1, by = 0.01))

## Preprocess training data
nearZeroVar(X.train)  ## determine which variables have constant variance
preProcValues <- preProcess(X.train, method = c("center", "scale"))
trainTransformed <- predict(preProcValues, X.train)

## Enet biomarker panel
enetFit <- train(x = trainTransformed,
  y = y.train,
  method = "glmnet",
  trControl = enetControl,
  tuneGrid = enetGrid,
  metric = "ROC")
enetFit
plot(enetFit)

## ROC was used to select the optimal model using  the largest value.
# The final values used for the model were alpha = 0.9 and lambda = 0.001.

#getting the coefficients of the final model
coefficients <- coef(enetFit$finalModel, enetFit$bestTune$lambda)
# miRNA biomarker panel
enetPanel <- names(which(as.matrix(coefficients)[-1, ] != 0))
length(enetPanel) # 34 features
```

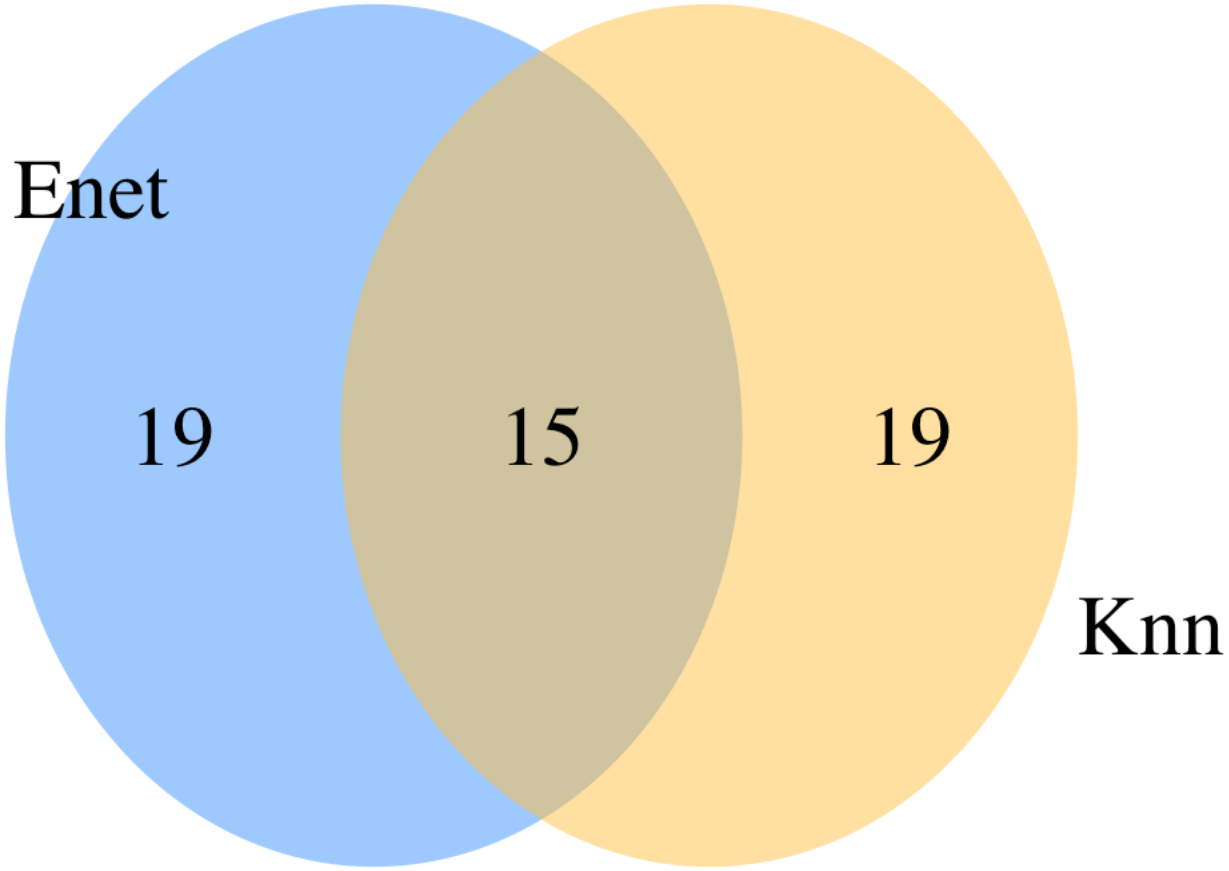


2) Determine biomarker panel using Knn (Knn)

```
## knn algorithm
## (applying univariate filtering using anova)
## keep the number of transcripts equal to the
## elastic net biomarker panel
mySBF <- caretSBF
mySBF$filter <- function(score, x, y)
                rank(score) <= length(enetPanel)

## set options for tuning parameters
set.seed(12)
knnControl <- trainControl(method = "repeatedcv",
  number = k, repeats = M,
  classProbs = TRUE,
  summaryFunction = twoClassSummary)
## set options for selection by filtering
set.seed(13)
sbfControl = sbfControl(functions = mySBF,
  method = "cv",
  number = k)
knnGrid = data.frame(k = c(5, 10, 20, 40))

## Build knn biomarker panel
knnFit <- sbf(x = trainTransformed,
  y = y.train,
  method = "knn",
  tuneGrid = knnGrid,
  sbfControl = sbfControl,
  trControl = knnControl,
  metric = "ROC")

# ROC was used to select the optimal model using  the largest value.
# The final value used for the model was k = 20.
knnFit$fit
knnFit$optVariables
```

```
k-Nearest Neighbors

105 samples
 34 predictor
  2 classes: 'Her2', 'LumA'

No pre-processing
Resampling: Cross-Validated (2 fold, repeated 2 times)
Summary of sample sizes: 53, 52, 52, 53
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   5  0.9261083  0.7333333  0.9665718
  10  0.9511202  0.7166667  0.9665718
  20  0.9581792  0.6000000  0.9868421
  40  0.9394678  0.0000000  1.0000000

ROC was used to select the optimal model using  the largest value.
The final value used for the model was k = 20.
```

Overlap between the Enet and Knn biomarker panels



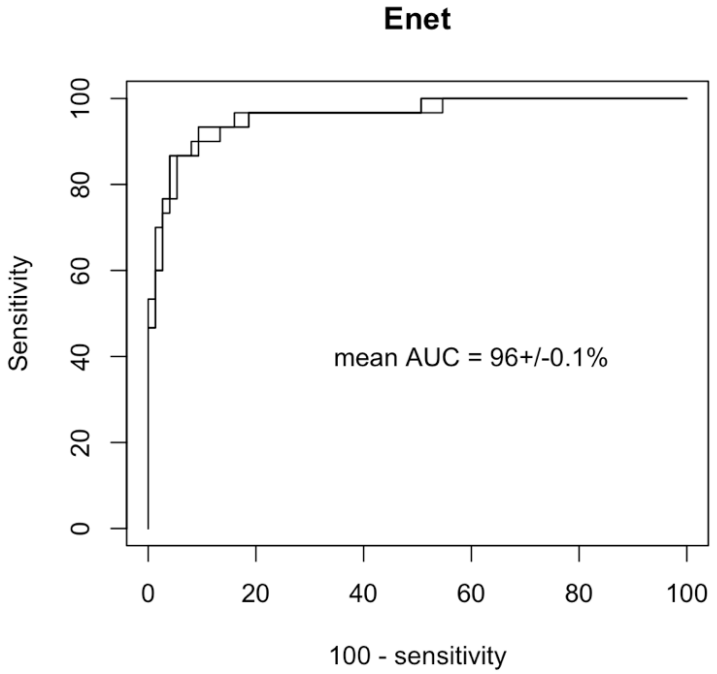
Estimate test performance

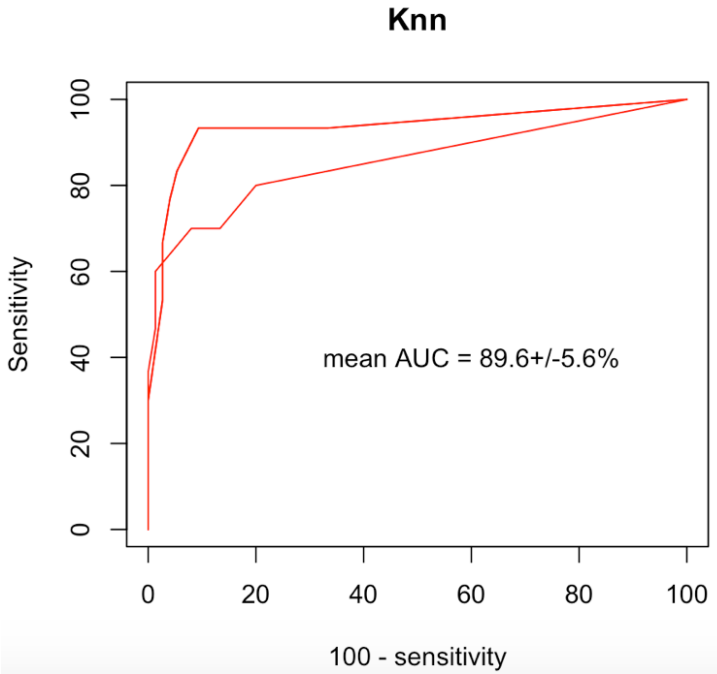
```r
## Perform repeatedCV to estimate test performance
set.seed(14)
# list of lists containing fold ids
folds <- lapply(1:M, function(i) createFolds(y.train, k = k))

probHer2.estimateTestPerf.enet <- lapply(folds, function(i){
  lapply(i, function(j){
    # tune parameters using cross-validation
    set.seed(15)
    enetControl <- trainControl(method = "repeatedcv",
      number = k, repeats = M,
      classProbs = TRUE,
      summaryFunction = twoClassSummary)
    enetGrid = expand.grid(alpha = c(0.9, 1),
      lambda = seq(0.001, 0.1, by = 0.01))

    ## Pre-processing data
    preProcValues <- preProcess(X.train[-j, ],
      method = c("center", "scale"))
    trainTransformed <- predict(preProcValues, X.train[-j, ])
    testTransformed <- predict(preProcValues, X.train[j, ])

    ## build elastic net classifier
    enetFit <- train(x = trainTransformed,
      y = y.train[-j],
      method = "glmnet",
      trControl = enetControl,
      tuneGrid = enetGrid,
      metric = "ROC")

    ## Estimate probabilities of test subjects
    probTest <- predict(enetFit, testTransformed, type = "prob")
    her2Prob <- probTest[, "Her2"]
    her2Prob
  })
})
```

```r
## estimate test performance of knn classifier
probHer2.estimateTestPerf.knn <- lapply(folds, function(i){
  lapply(i, function(j){
    # set options
    mySBF <- caretSBF
    mySBF$filter <- function(score, x, y)
      rank(score) <= length(enetPanel)

    set.seed(16)
    # tuning paramters using cross-validation
    knnControl <- trainControl(method = "repeatedcv",
      number = k, repeats = M,
      classProbs = TRUE,
      summaryFunction = twoClassSummary)
    # selection by filtering
    set.seed(17)
    sbfControl = sbfControl(functions = mySBF,
      method = "cv",
      number = k)
    knnGrid = data.frame(k = c(5, 10, 20, 40))

    ## Pre-processing data
    preProcValues <- preProcess(X.train[-j, ],
      method = c("center", "scale"))
    trainTransformed <- predict(preProcValues, X.train[-j, ])
    testTransformed <- predict(preProcValues, X.train[j, ])

    ## build knn classifier
    knnFit <- sbf(x = trainTransformed,
      y = y.train[-j],
      method = "knn",
      tuneGrid = knnGrid,
      sbfControl = sbfControl,
      trControl = knnControl,
      metric = "ROC")

    ## Estimate probabilities of test subjects
    probTest <- predict(knnFit, testTransformed, type = "prob")
    her2Prob <- probTest[, "Her2"]
    her2Prob
  })
})
```

Enet biomarker panel performs better than the Knn biomarker panel

```r
## Compute classification performance measures
## Enet
trainPerfEnet <- mapply(function(x, y){
  auc <- pROC::roc(y.train[unlist(x)], unlist(y),
    direction = "<",
    levels = c("LumA", "Her2"),
    percent=TRUE)
  list(tpr = auc$sensitivities,
    fpr = 100-auc$specificities,
    auc =  round(auc$auc, 2))
}, x = folds, y = probHer2.estimateTestPerf.enet)

## Knn
trainPerfKnn <- mapply(function(x, y){
  auc <- pROC::roc(y.train[unlist(x)], unlist(y),
    direction = "<",
    levels = c("LumA", "Her2"),
    percent=TRUE)
  list(tpr = auc$sensitivities,
    fpr = 100-auc$specificities,
    auc =  round(auc$auc, 2))
}, x = folds, y = probHer2.estimateTestPerf.knn)

## plot ROC curves
par(mfrow = c(2, 1))
plot(trainPerfEnet["tpr", ][[1]] ~ trainPerfEnet["fpr", ][[1]],
  type = "l", col = 1, xlab = "100 - sensitivity",
  ylab = "Sensitivity", main = "Enet")
for(i in 1:length(folds)){
  points(trainPerfEnet["tpr", ][[i]] ~ trainPerfEnet["fpr", ][[i]],
    type = "l", col = 1)
}
text(x = 60, y = 40, labels =
    paste0("mean AUC = ", round(mean(unlist(trainPerfEnet["auc", ])), 1),
      "+/-", round(sd(unlist(trainPerfEnet["auc", ])), 1), "%"))
plot(trainPerfKnn["tpr", ][[1]] ~ trainPerfKnn["fpr", ][[1]],
  type = "l", col = 2, xlab = "100 - sensitivity",
  ylab = "Sensitivity", main = "Knn")
for(i in 1:length(folds)){
  points(trainPerfKnn["tpr", ][[i]] ~ trainPerfKnn["fpr", ][[i]],
    type = "l", col = 2)
}
text(x = 60, y = 40, labels =
    paste0("mean AUC = ", round(mean(unlist(trainPerfKnn["auc", ])), 1),
      "+/-", round(sd(unlist(trainPerfKnn["auc", ])), 1), "%"))
```





Conclusions

- There are many methods and algorithms to perform classification and regression (more than those covered here).
- There are many variable selection methods (more than those covered here).
- Which one is better? We can evaluate different options using cross-validation.
- There is a trade-off between bias and variance when we fit more complex models.
- Even good CV performance does not mean that you will get good performance in the test set.
- LASSO and Ridge are particular cases of EN. The latter may perform better if covariates are highly correlated (group effect).
- Other extensions of LASSO to solve the grouping problem have been proposed.