

Penalized Regression

Max Turgeon

STAT 7200–Multivariate Statistics

Objectives

- Introduce ridge regression and discuss the bias-variance trade-off
- Introduce Lasso regression and discuss variable selection
- Discuss cross-validation for parameter tuning

Recall: Least Squares Estimation i

- Let $\mathbf{Y}_1, \dots, \mathbf{Y}_n$ be a random sample of size n , and let $\mathbf{X}_1, \dots, \mathbf{X}_n$ be the corresponding sample of covariates.
 - \mathbf{Y}_i and \mathbf{X}_i are of dimension p and q , respectively.
- We will write \mathbb{Y} and \mathbb{X} for the matrices whose i -th row is \mathbf{Y}_i and \mathbf{X}_i , respectively.
- From the linear model assumption, we can then write $E(\mathbb{Y} \mid \mathbb{X}) = \mathbb{X}B$.
- The least-squares criterion is given by

$$LS(B) = \text{tr} \left[(\mathbb{Y} - \mathbb{X}B)^T (\mathbb{Y} - \mathbb{X}B) \right].$$

Recall: Least Squares Estimation ii

- The minimum is attained at at

$$\hat{B} = (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T \mathbb{Y}.$$

- The least-squares estimator is *unbiased*:

$$E(\hat{B} \mid \mathbb{X}) = B.$$

- If we let $\hat{\beta}_i$ be the i -th column of \hat{B} , we have

$$\text{Cov}(\hat{\beta}_i, \hat{\beta}_j) = \sigma_{ij} (\mathbb{X}^T \mathbb{X})^{-1},$$

where σ_{ij} is the (i, j) -th entry of $\Sigma = \text{Cov}(\mathbf{Y}_i \mid \mathbf{X}_i)$.

Multicollinearity

- As we can see, the variance of the regression coefficients depend on the inverse of $\mathbb{X}^T \mathbb{X}$.
- **Multicollinearity** is when the columns of \mathbb{X} are *almost* linearly dependent.
 - Note: This can happen when a covariate is almost constant.
- As a consequence, $\mathbb{X}^T \mathbb{X}$ is nearly singular, and therefore the variance of the regression coefficients can blow up.

- **Solution:** Add a small positive quantity along the diagonal of $\mathbb{X}^T \mathbb{X}$.
 - $\mathbb{X}^T \mathbb{X} \rightarrow \mathbb{X}^T \mathbb{X} + \lambda I$
- The **Ridge estimator** of B is given by

$$\hat{B}_R = (\mathbb{X}^T \mathbb{X} + \lambda I_q)^{-1} \mathbb{X}^T \mathbb{Y}.$$

Example i

```
library(tidyverse)
url <- "https://maxturgeon.ca/w20-stat7200/prostate.csv"
prostate <- read_csv(url)

# Separate into training and testing sets
data_train <- filter(prostate, train == TRUE) %>%
  dplyr::select(-train)
data_test <- filter(prostate, train == FALSE) %>%
  dplyr::select(-train)
```

Example ii

```
# OLS
model1 <- lm(lpsa ~ .,
             data = data_train)
pred1 <- predict(model1, data_test)

mean((data_test$lpsa - pred1)^2)

## [1] 0.521274
```


Example iii

```
# Ridge regression
X_train <- model.matrix(lpsa ~ .,
                        data = data_train)
Y_train <- data_train$lpsa

B_ridge <- solve(crossprod(X_train) + diag(0.7, 9),
                 t(X_train)) %*% Y_train
```

Example iv

```
X_test <- model.matrix(lpsa ~ .,  
                        data = data_test)  
  
pred2 <- X_test %*% B_ridge  
  
mean((data_test$lpsa - pred2)^2)  
  
## [1] 0.5180924  
  
# Compare both estimates  
head(cbind(coef(model1), B_ridge))
```

Example v

```
##                [,1]        [,2]
## (Intercept)  0.42917013  0.1323063
## lcavol      0.57654319  0.5709660
## lweight     0.61402000  0.6160020
## age        -0.01900102 -0.0173843
## lbph       0.14484808  0.1395858
## svi        0.73720864  0.6683160
```

Bias-Variance tradeoff i

- The ridge estimator is **biased**:

$$\begin{aligned}E(\hat{B}_R | \mathbb{X}) &= (\mathbb{X}^T \mathbb{X} + \lambda I_q)^{-1} \mathbb{X} E(\mathbb{Y} | \mathbb{X}) \\ &= (\mathbb{X}^T \mathbb{X} + \lambda I_q)^{-1} \mathbb{X}^T \mathbb{X} B \\ &\neq B.\end{aligned}$$

- But the variance is potentially smaller:

$$\text{Cov}(\hat{\beta}_i, \hat{\beta}_j) = \sigma_{ij} (\mathbb{X}^T \mathbb{X} + \lambda I_q)^{-1} \mathbb{X}^T \mathbb{X} (\mathbb{X}^T \mathbb{X} + \lambda I_q)^{-1}.$$

- This is an example of the classical **bias-variance tradeoff**:
 - We increase bias and decrease variance.

- Ideally, this is done in such a way to reduce the **mean squared error**:

$$MSE = \frac{1}{n} \text{tr} \left[(\mathbf{Y} - \hat{\mathbf{Y}})^T (\mathbf{Y} - \hat{\mathbf{Y}}) \right].$$

- *Should we compute the MSE with the training or the test data?*

Example (cont'd) i

```
mse_df <- purrr::map_df(seq(0, 5, by = 0.1),
                        function(lambda) {
  B_ridge <- solve(crossprod(X_train) + diag(lambda, 9),
                  t(X_train)) %*% Y_train
  pred2 <- X_test %*% B_ridge

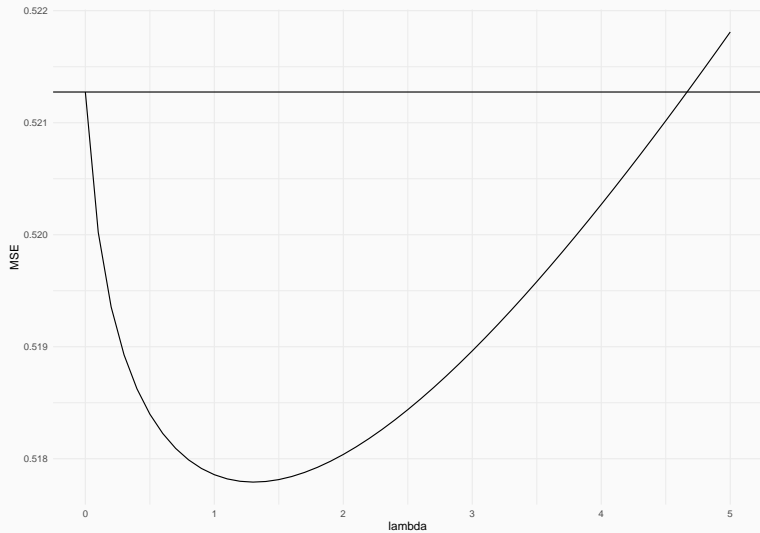
  mse <- mean((data_test$lpsa - pred2)^2)
  return(data.frame(MSE = mse,
                    lambda = lambda))
})
```

Example (cont'd) ii

```
ols_mse <- mean((data_test$lpsa - pred1)^2)
```

```
ggplot(mse_df, aes(lambda, MSE)) +  
  geom_line() + theme_minimal() +  
  geom_hline(yintercept = ols_mse)
```

Example (cont'd) iii



Regularized regression

- The ridge estimator can also be defined as a solution to a **regularized least squares problem**:

$$LS_R(B; \lambda) = \text{tr} \left[(\mathbb{Y} - \mathbb{X}B)^T (\mathbb{Y} - \mathbb{X}B) \right] + \lambda \text{tr} (B^T B).$$

- Yet another way to define the ridge estimator is as a solution to a **constrained least squares problem**:

$$\min_B \text{tr} \left[(\mathbb{Y} - \mathbb{X}B)^T (\mathbb{Y} - \mathbb{X}B) \right], \quad \text{tr} (B^T B) \leq c.$$

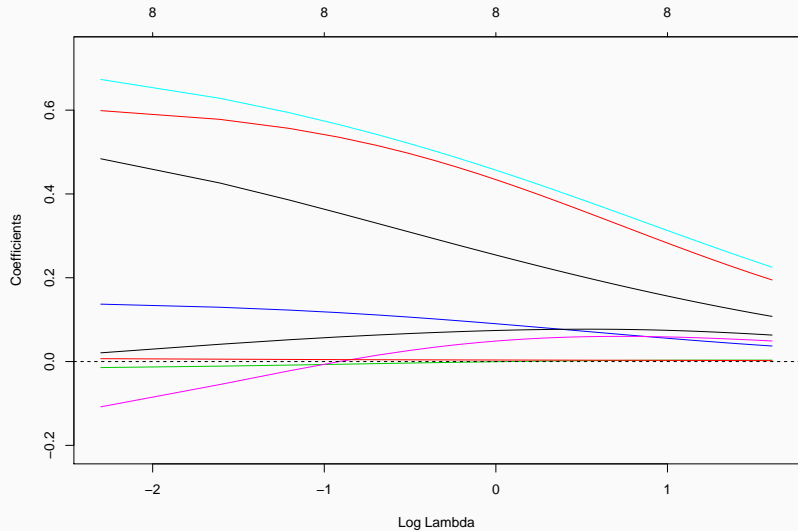
```
library(glmnet)

# Fit for multiple values of lambda
X_train <- model.matrix(lpsa ~ . - 1,
                       data = data_train)
ridge_fit <- glmnet(X_train, data_train$lpsa,
                   alpha = 0,
                   lambda = seq(0, 5, by = 0.1))
```

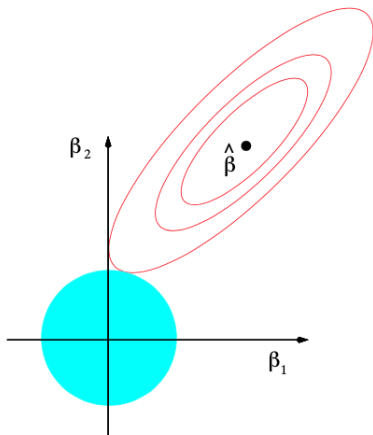
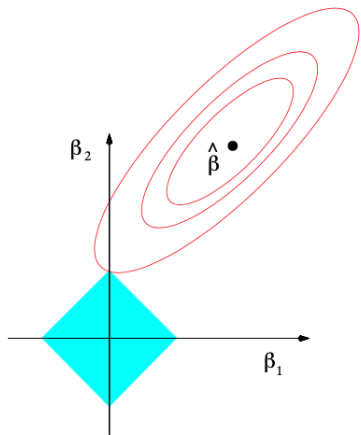
Solution path ii

```
# Plot the value of the coefficients  
# as a function of lambda  
plot(ridge_fit, xvar = "lambda")  
abline(h = 0, lty = 2)
```

Solution path iii



Constrained regression



Lasso regression

- **Lasso regression** puts a different constraint on the size of the regression coefficients B :
 - Ridge regression: $\text{tr}(B^T B) = \sum_{ij} B_{ij}^2 \leq c$
 - Lasso regression: $\|B\|_1 = \sum_{ij} |B_{ij}| \leq c$
- Just as with ridge regression, this is also equivalent to a regularized least squares problem:

$$LS_L(B; \lambda) = \text{tr} \left[(\mathbb{Y} - \mathbb{X}B)^T (\mathbb{Y} - \mathbb{X}B) \right] + \lambda \|B\|_1.$$

- **Major difference:** Lasso regression performs *variable selection*.

Example (cont'd) i

```
# Fit lasso regression along the same lambda sequence
lasso_fit <- glmnet(X_train, data_train$lpsa,
                   alpha = 1, # For lasso regression
                   lambda = seq(0, 5, by = 0.1))

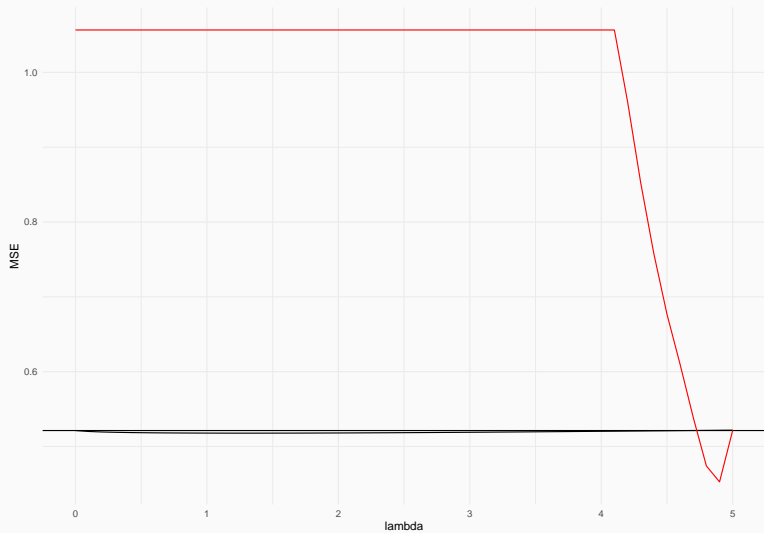
X_test <- model.matrix(lpsa ~ . - 1,
                      data = data_test)

lasso_pred <- predict(lasso_fit, newx = X_test)
lasso_mse <- apply(lasso_pred, 2, function(col) {
  mean((data_test$lpsa - col)^2)
})
```

Example (cont'd) ii

```
lasso_mse_df <- data.frame(MSE = lasso_mse,  
                           lambda = seq(0, 5, by = 0.1))  
ggplot(mse_df, aes(lambda, MSE)) +  
  geom_line() + theme_minimal() +  
  geom_hline(yintercept = ols_mse) +  
  geom_line(data = lasso_mse_df, colour = 'red')
```

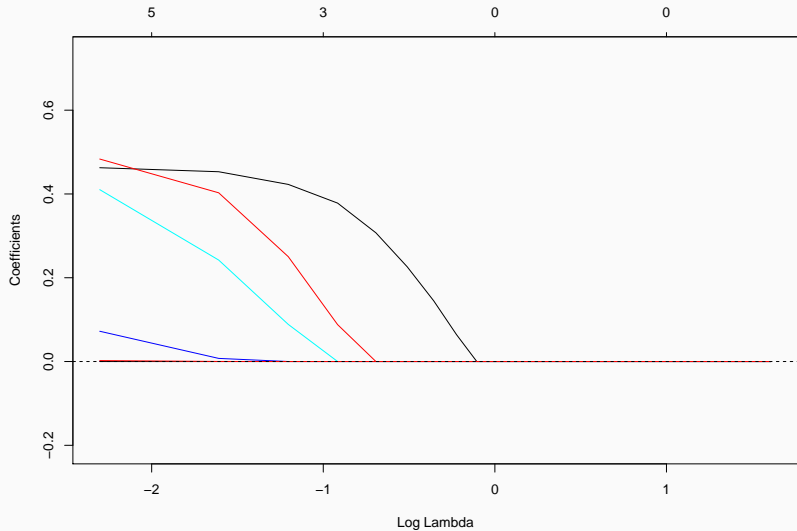

Example (cont'd) iii



Example (cont'd) iv

```
# Plot the value of the coefficients  
# as a function of lambda  
plot(lasso_fit, xvar = "lambda")  
abline(h = 0, lty = 2)
```

Example (cont'd) v



Example (cont'd) vi

```
# Where is the min MSE?
```

```
filter(lasso_mse_df, MSE == min(MSE))
```

```
##           MSE lambda
```

```
## 1 0.4526232    4.9
```

```
# What are the estimates?
```

```
coef(lasso_fit, s = 4.9)
```

Example (cont'd) vii

```
## 9 x 1 sparse Matrix of class "dgCMatrix"  
##                1  
## (Intercept) 2.452345  
## lcavol      .  
## lweight     .  
## age         .  
## lbph        .  
## svi         .  
## lcp         .  
## gleason     .  
## pgg45       .
```

- There are other forms of penalized regression:
 - Elastic net, SCAD, adaptive lasso, group lasso, etc.
- They each have different *asymptotic* and *finite sample* properties.
 - E.g. Lasso is asymptotically biased; Elastic net and SCAD are asymptotically unbiased.
- In general, how do we select λ when we don't have a test set?
 - **Answer:** Cross-validation.

K-fold cross-validation

- Goal: Find the value of λ that minimises the MSE on test data.
- K -fold cross-validation (CV) is a resampling technique that estimates the test error from the training data.
- It is also an efficient way to use all your data, as opposed to separating your data into a training and a testing subset.

Algorithm

Let $K > 1$ be a positive integer.

1. Separate your data into K subsets of (approximately) equal size.
2. For $k = 1, \dots, K$, put aside the k -th subset and use the remaining $K - 1$ subsets to train your algorithm.
3. Using the trained algorithm, predict the values for the held out data.
4. Calculate MSE_k as the Mean Squared Error for these predictions.
5. The overall MSE estimate is given by

$$MSE = \frac{1}{K} \sum_{k=1}^K MSE_k.$$

Example i

```
# Take all the data
```

```
dataset <- dplyr::select(prostate, -train)  
dim(dataset)
```

```
## [1] 97  9
```

```
set.seed(7200)
```

```
library(caret)
```

```
# 5-fold CV
```

```
trainIndex <- createFolds(dataset$lpsa, k = 5)  
str(trainIndex)
```

Example ii

```
## List of 5
## $ Fold1: int [1:20] 6 8 22 23 25 27 28 32 41 46 ...
## $ Fold2: int [1:19] 5 7 15 18 20 26 29 42 44 45 ...
## $ Fold3: int [1:19] 1 11 19 21 24 30 33 48 49 50 ...
## $ Fold4: int [1:19] 3 4 10 12 16 31 34 35 39 43 ...
## $ Fold5: int [1:20] 2 9 13 14 17 36 37 38 40 47 ...
```

Example iii

```
# Define function to compute MSE
compute_mse <- function(prediction, actual) {
  # Recall: the prediction comes in an array
  apply(prediction, 2, function(col) {
    mean((actual - col)^2)
  })
}
```

Example iv

```
MSEs <- sapply(trainIndex, function(indices){
  X_train <- model.matrix(lpsa ~ . - 1,
                          data = dataset[-indices,])
  Y_train <- dataset$lpsa[-indices]
  X_test <- model.matrix(lpsa ~ . - 1,
                         data = dataset[indices,])
  lasso_fit <- glmnet(X_train, Y_train, alpha = 1,
                     lambda = seq(0, 5, by = 0.1))
  lasso_pred <- predict(lasso_fit, newx = X_test)
  compute_mse(lasso_pred, dataset$lpsa[indices])
})
```

Example v

```
# Each column is for a different fold
```

```
dim(MSEs)
```

```
## [1] 51  5
```

```
CV_MSE <- colMeans(MSEs)
```

```
seq(0, 5, by = 0.1)[which.min(CV_MSE)]
```

```
## [1] 0.4
```

Example vi

```
# What are the estimates?
```

```
coef(lasso_fit, s = 0.4)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                1
```

```
## (Intercept) 1.63646053
```

```
## lcavol      0.37816202
```

```
## lweight    0.08802054
```

```
## age        .
```

```
## lbph       .
```

```
## svi        .
```

Example vii

```
## lcp      .  
## gleason .  
## pgg45   .  
  
# Conveniently, glmnet has a function for CV  
# It also chooses the lambda sequence for you  
X <- model.matrix(lpsa ~ . -1, data = dataset)  
lasso_cv_fit <- cv.glmnet(X, dataset$lpsa, alpha = 1,  
                          nfolds = 5)  
  
c("lambda.min" = lasso_cv_fit$lambda.min,  
  "lambda.1se" = lasso_cv_fit$lambda.1se)
```

Example viii

```
## lambda.min lambda.1se  
## 0.03250172 0.14400281
```

What are the estimates?

```
coef(lasso_cv_fit, s = 'lambda.min')
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"  
##                1  
## (Intercept)  0.161190494  
## lcavol      0.508157223  
## lweight     0.552486889  
## age        -0.009374709
```


Example ix

```
## lbph          0.064736544  
## svi          0.594693519  
## lcp          .  
## gleason      0.004797184  
## pgg45        0.002351087
```

```
# 1 SE rule
```

```
coef(lasso_cv_fit, s = 'lambda.1se')
```

Example x

```
## 9 x 1 sparse Matrix of class "dgCMatrix"  
##                1  
## (Intercept) 0.2975535  
## lcavol      0.4725492  
## lweight     0.3989087  
## age         .  
## lbph        .  
## svi         0.4400593  
## lcp         .  
## gleason     .  
## pgg45       .
```

Summary

- Regularized regression can help reduce the mean-squared error, especially in the presence of multicollinearity
 - **Ridge regression:** Penalizes the $L2$ norm of the coefficients
 - **Lasso regression:** Penalizes the $L1$ norm of the coefficients
- Unlike ridge regression, lasso regression also performs variable selection.
 - But this comes at a cost: **post-selection inference**.
- K -fold cross-validation can be used to find the best value of λ .