Regularized Regression

Max Turgeon

DATA 2010–Tools and Techniques in Data Science

Lecture Objectives

- Describe the concept of regularized regression.
- Compare and contrast ridge and lasso regression.

Motivation

- In the previous lecture, we discussed two different strategies for improving accuracy of a linear regression model.
    - Add extra covariates.
    - Model continuous covariates non-linearly.
- **Regularization** is another approach.
- It is based on two concepts:
    - Mitigate the impact of outliers.
    - Leverage the bias-variance tradeoff.

Recall: Least-Squares Estimation i

- Let $Y_1, \ldots, Y_n$ be a random sample of size n, and let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be the corresponding sample of covariates.
- We will write $\mathbb{Y}$ for the vector whose i-th element is Y_i, and $\mathbb{X}$ for the matrix whose i-th row is $\mathbf{X}_i$.
- The Least-Squares estimate $\hat{\beta}$ is given by

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

- **How do we get this formula?** By solving an optimization problem. Define the following function of β:

Recall: Least-Squares Estimation ii

$$L(\beta) = \frac{1}{2n} \sum_{i=1}^{n} \left(Y_i - \beta^T \mathbf{X}_i \right)^2 .$$

- Using Calculus, you can show that the value of β which *minimizes* $L(\beta)$ is exactly $\hat{\beta}$.

- When there a lot of features (i.e. $p \approx n$) or when there is high correlation (e.g. image data), the least square estimate is ill-behaved.
- **Why?** Because we need to invert the near-singular matrix $\mathbb{X}^T\mathbb{X}$.
    - This leads to very large estimates and high variance.
- Hoerl & Kennard (1970) suggested adding a fix quantity $\lambda > 0$ to the diagonal of $\mathbb{X}^T\mathbb{X}$ before inverting:

$$\hat{\beta}_\lambda = (\mathbb{X}^T\mathbb{X} + \lambda I_p)^{-1}\mathbb{X}^T\mathbb{Y}.$$

Ridge Regression ii

- Their estimate is biased, but it has lower variance than OLS when $p \approx n$.
- The constant λ is known as a **hyper-parameter**. Different values will lead to different performance, and ideally we would choose λ with the best performance.
  - We will come back to this idea in a future lecture.

Example i

```
library(tidyverse)
url <- paste0("https://web.stanford.edu/~hastie/",
              "ElemStatLearn/datasets/prostate.data")
dataset <- read.table(url)
```

```
# Separate train and test
data_train <- filter(dataset, train)
data_test <- filter(dataset, !train)
```

Example ii

```r
# Model without gleason
fit <- lm(lpsa ~ lcavol + lweight + age + lbph +
            svi + lcp + pgg45, data = data_train)
coef(fit)
```

```
## (Intercept) lcavol lweight age lbph svi
## 0.259061747 0.573930391 0.619208833
-0.019479879 0.144426474 0.741781258
## lcp pgg45
## -0.205416986 0.008944996
```

Example iii

```r
# For ridge regression, we need the model matrix
# and the vector y
X <- model.matrix(fit)
y <- data_train$lpsa
```

```r
beta_ols <- solve(crossprod(X)) %*%
  crossprod(X, y)
all.equal(coef(fit), beta_ols[,1])
```

```
## [1] TRUE
```

Example iv

```r
lambda <- 1.0
p <- ncol(X)
beta_ridge <- solve(crossprod(X) + diag(lambda,
                                        ncol = p,
                                        nrow = p)) %*%
  crossprod(X, y)

beta_ridge[,1]
```

Example v

```
## (Intercept) lcavol lweight age lbph svi
## 0.125969111 0.570043249 0.611735434
-0.016497853 0.138181026 0.640871861
## lcp pgg45
## -0.183445103 0.008872163
```

Exercise

Compute the RMSE of the ridge estimate when $\lambda = 1$.

Hint: You'll need to create a matrix X_{test} and get the predicted values using matrix multiplication. Use `model.frame(formula, data = data_test)` and convert the output to a matrix.

Solution i

```r
# Use the same formula as for fitting the model
X_test <- model.matrix(~ lcavol + lweight + age +
                          lbph + svi + lcp + pgg45,
                       data = data_test)
X_test <- as.matrix(X_test)
dim(X_test)
```

```
## [1] 30  8
```

Solution ii

```
y_test <- data_test$lpsa
y_pred <- X_test %*% beta_ridge

rmse <- sqrt(mean((y_test - y_pred)^2))
rmse
```

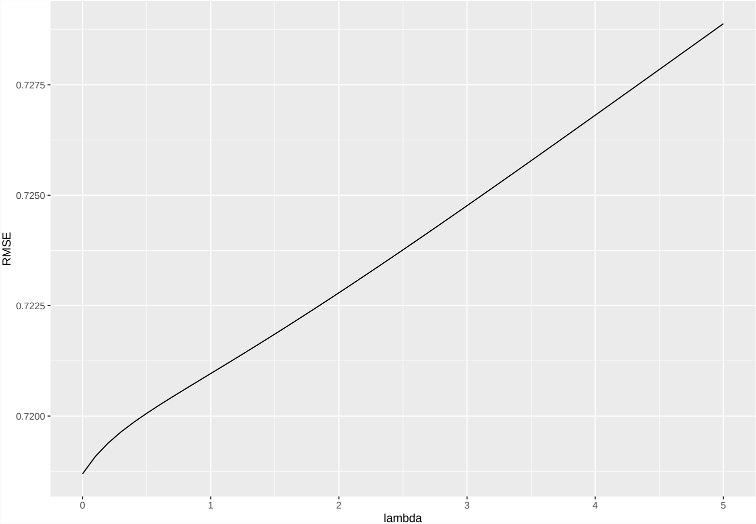
```
## [1] 0.7209613
```

```
# Compare to OLS estimate
pred_vals <- predict(fit, newdata = data_test)
sqrt(mean((y_test - pred_vals)^2))
```

Solution iii

```
## [1] 0.7186887
```

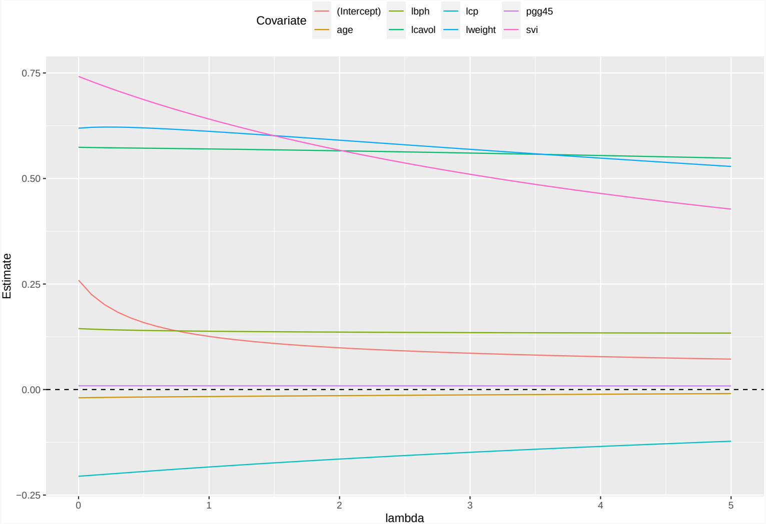
Effect of λ i



Effect of λ ii

- In this particular example, as we increase λ, we don't get a better RMSE.
    - In other words, OLS has better performance than ridge regression of these covariates.
- But what we typically see is that RMSE will decrease, but only up to a point: as λ becomes larger and larger, the increase in bias overpowers the reduction in variance, and the overall performance suffers.

Effect of λ iii



Effect of λ iv

- As we increase λ, the estimates *shrink*: they get closer to 0 in absolute value.
- In fact, we can show that as $\lambda \to \infty$, we have $\beta \to 0$.

Ridge regression as regularized regression i

- There are two ways to justify the ridge regression estimate:
    - As constrained optimization.
    - As regularized (or penalized) estimation.
- **Constrained**: Minimize $L(\beta)$ subject to $\|\beta\|_2 \leq \mu$ for some $\mu > 0$.
    - This explains the shrinkage.
- **Regularized**: We change $L(\beta)$ to

$$L_{Ridge}(\beta; \lambda) = \frac{1}{2n} \sum_{i=1}^{n} \left(Y_i - \beta^T \mathbf{X}_i\right)^2 + \lambda \|\beta\|_2^2.$$

Ridge regression as regularized regression ii

- The two approaches are equivalent.
    - There is a one-to-one correspondence between λ and μ.
- The constrained view is explicit about the goal: keep the estimates small.
- The regularized view is a trade-off: you want to minimize the Least-Squares criterion, while controlling the size of the estimates.

Lasso regression

- Ridge regression can be generalized in a number of ways. The most popular approach is **Lasso regression**:

$$L_{Lasso}(\beta;\lambda) = \frac{1}{2n}\sum_{i=1}^{n}\left(Y_i - \beta^T\mathbf{X}_i\right)^2 + \lambda\|\beta\|_1.$$

- *Note*: We have $\|\beta\|_1 = \sum_{k=0}^{p}|\beta_k|$.
- Some comments:
    - In general, there is no closed-form solution to the Lasso optimization problem.
    - $L_{Lasso}(\beta;\lambda)$ is not differentiable with respect to β, and therefore we can't use standard algorithms (e.g. Gradient descent, Newton-Raphson) to find a solution.
    - Lasso regression actually performs **variable selection**.

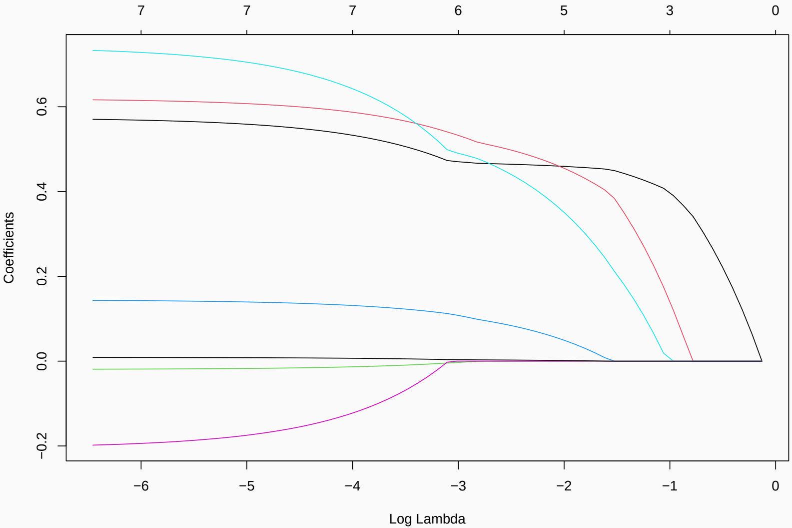
Example i

```r
library(glmnet)
# We need to remove the intercept from X
X <- X[, -1]

fit_lasso <- glmnet(X, y)
```

```r
# Note: The x-axis is on the log scale
plot(fit_lasso, xvar = "lambda")
```

Example ii



Example iii

```r
# Use predict to get predictions
X_test <- X_test[, -1]
pred_lasso <- predict(fit_lasso, newx = X_test,
                      s = 1.0) # s = lambda

rmse_lasso <- sqrt(mean((y_test - pred_lasso)^2))
rmse_lasso
```

```
## [1] 1.027975
```

Exercise

Compute the RMSE of lasso regression for `s = 0.11`.

Solution

```r
pred_lasso <- predict(fit_lasso, newx = X_test,
                      s = 0.11) # s = lambda

sqrt(mean((y_test - pred_lasso)^2))
```

```
## [1] 0.6725187
```

Final remarks

- Ridge and lasso regression are examples of regularized linear regression.
    - The same ideas can be used for other forms of regression (e.g. logistic regression).
- They can be combined, giving rise to **elastic-net regression**.
    - It performs variable selection (like lasso) but has better theoretical properties (e.g. consistency).
- Regularization is a very important topic in modern data science.
    - Neural networks are frequently combined with ridge and/or lasso ideas to avoid over-fitting.