Generalisation

Anastasia Chanbour

29/10/2021

Loading the required packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(broom)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-3
```

High-dimensional regression

Generating some data from a high-dimensional model (increasing p below)

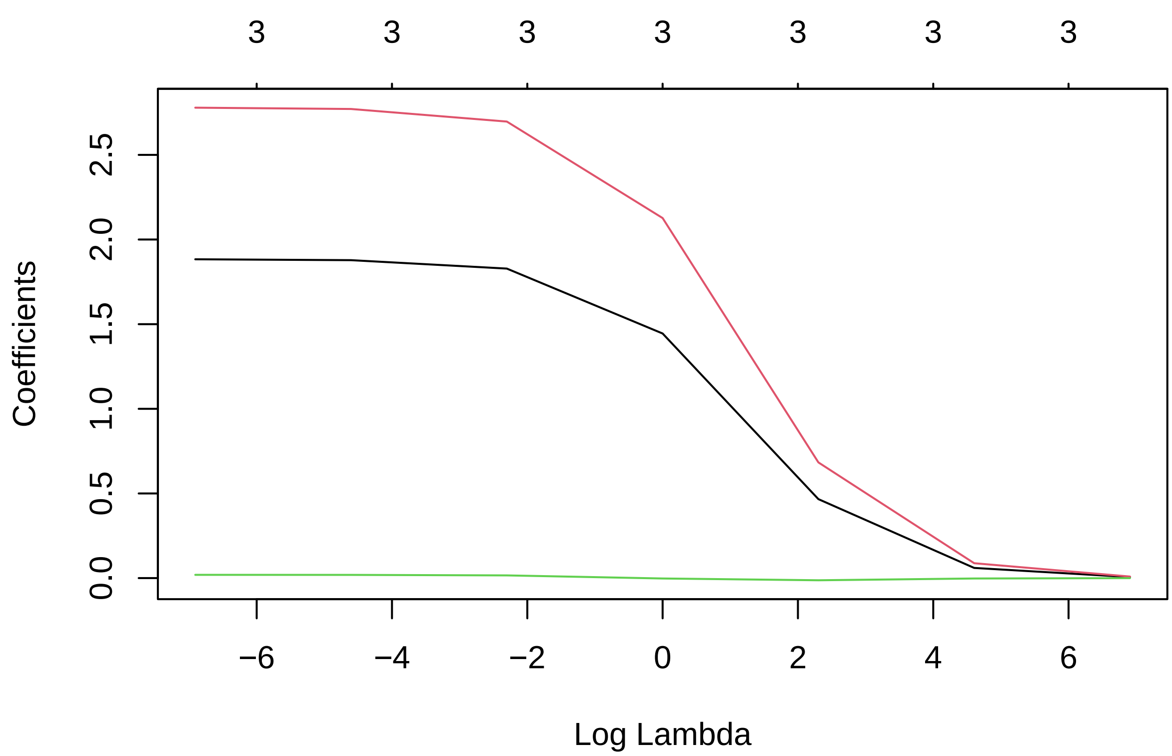
```
set.seed(1)
n <- 100
p <- 3
X <- matrix(rnorm(n*p), nrow = n)
beta <- rpois(p, lambda = 1)
y <- X %*% beta + rnorm(n)
```

Ridge regression with glmnet

```
model_ridge <- glmnet(X, y,
                      intercept = FALSE,
```

```r
                      alpha =0,
                      lambda = 10^(-3:3))
```

```r
plot(model_ridge, xvar = "lambda")
```



Estimation error

Computing the mean-squared error of the coefficient estimates at different values of lambda

```r
lambda <- 10^0
beta_hat <- coef(model_ridge, s = lambda)[-1]# leave out intercept
mean((beta - beta_hat)^2)
```

```
## [1] 0.3569038
```

Prediction error

Computing predictions using the estimated coefficients and the mean-squared prediction error at different values of lambda

```r
# e.g. predict(model_ridge, newx = X, s = lambda or 10*lambda)
y_hat <- X%*%beta_hat

#mse prediction of yhat
mean((y-y_hat)^2)
```

```
## [1] 1.3833
```

Overfitting to variance and In Distribution (ID) generalisation

Generating a new sample from the same distribution (which things stay fixed?)

```
X_ID <- matrix(rnorm(n*p),nrow=n)
y_ID <- X_ID%*%beta +rnorm(n)
```

Calculating the prediction error on this new sample at different values of lambda

```
y_ID_hat <- predict(model_ridge, newx = X_ID, s = lambda )
#MSE
mean((y_ID - y_ID_hat)^2)
```

```
## [1] 1.879422
```

"Overfitting to bias" and Out Of Distribution (OOD) generalisation

There are many ways to change the distribution for a new sample

- Changing beta (adding a small amount of noise)
- Changing the CEF some other way (e.g. adding a non-linear term)
- Changing the distribution of X and/or the errors

We choose adding a noise to the beta coefficient

```
beta_new_D <- beta + rnorm(p, sd=0.1)
X_OOD <- matrix(rnorm(n*p),nrow=n)
y_OOD <- X_OOD %*% beta_new_D +rnorm(n)
```

```
y_OOD_hat <- predict(model_ridge, newx = X_OOD,s=lambda)

#MSE
mean((y_OOD - y_OOD_hat)^2)
```

```
## [1] 1.942832
```

Comparison with gradient descent

```
least_squares_gradient <- function(x, y, beta) {
  -2 * t(x) %*% (y - x %*% beta) #+ 2 * beta
}

least_squares_loss <- function(x, y, beta) {
  sum((y - x %*% beta)^2)
}
beta_prev2 <- rep(0, p) # or random start point
grad_prev2 <- least_squares_gradient(X, y, beta_prev2)
beta_prev1 <- beta_prev2 + 0.1 * grad_prev2 / sqrt(sum(grad_prev2^2))
grad_prev1 <- least_squares_gradient(X, y, beta_prev1)
previous_loss <- least_squares_loss(X, y, beta_prev2)
next_loss <- least_squares_loss(X, y, beta_prev1)
steps <- 1
while (abs(previous_loss - next_loss) > 0.001) {
  grad_diff <- grad_prev1 - grad_prev2
  step_BB <- sum((beta_prev1 - beta_prev2) * grad_diff) / sum(grad_diff^2)
  # Barzilai-Borwein step size
```

```r
  beta_prev2 <- beta_prev1
  beta_prev1 <- beta_prev1 - step_BB * grad_prev1

  grad_prev2 <- grad_prev1
  grad_prev1 <- least_squares_gradient(X, y, beta_prev1)

  previous_loss <- next_loss
  next_loss <- least_squares_loss(X, y, beta_prev1)

  print(previous_loss)
  steps <- steps + 1
}
```

```
## [1] 1128.978
## [1] 88.81974
## [1] 84.38873
## [1] 84.2721
## [1] 84.27054
```

```r
beta_final <- beta_prev1
```