Ridge, LASSO and Smoothing

2023-03-02

Task 1

For this task we use the `mogavs` dataset on communities and crime:

```
library(mogavs)
data("crimeData")
```

We normalise and divide our data into training and testing sets:

```
crimeData <- scale(crimeData)

train <- sample(1:nrow(crimeData), 1500)
X_train <- crimeData[train, -123]
y_train <- crimeData[train, 123]

X_test <- crimeData[-train, -123]
y_test <- crimeData[-train, 123]
```

Plotting the Paths

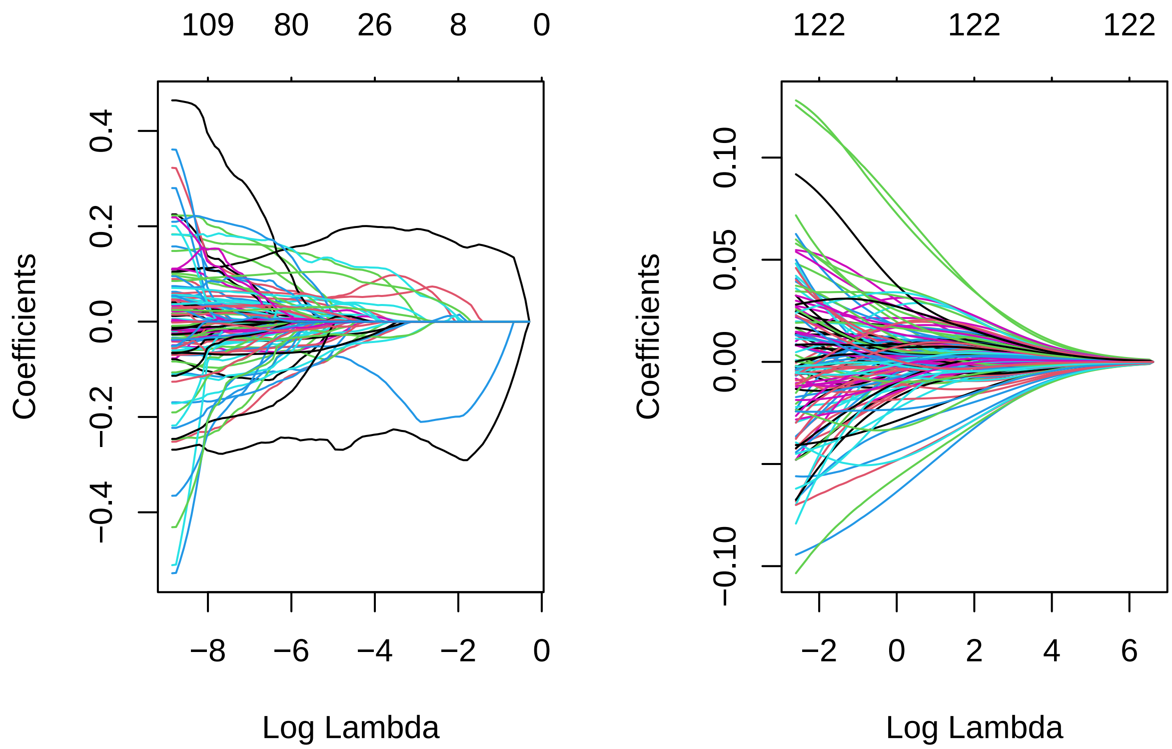
We use the `glmnet` package to perform LASSO, setting the `alpha` parameter to be 1 and ridge regression, by using `alpha = 0`. This gives us the following path visualisations, with the left plot showing the LASSO regression and the right plot showing the ridge regression:

```
library(glmnet)

lasso_fit <- glmnet(X_train, y_train, family = "gaussian", alpha = 1)


ridge_fit <- glmnet(X_train, y_train, family = "gaussian", alpha = 0)


par(mfrow = c(1,2))
plot(lasso_fit, xvar = "lambda")
plot(ridge_fit, xvar = "lambda")
```



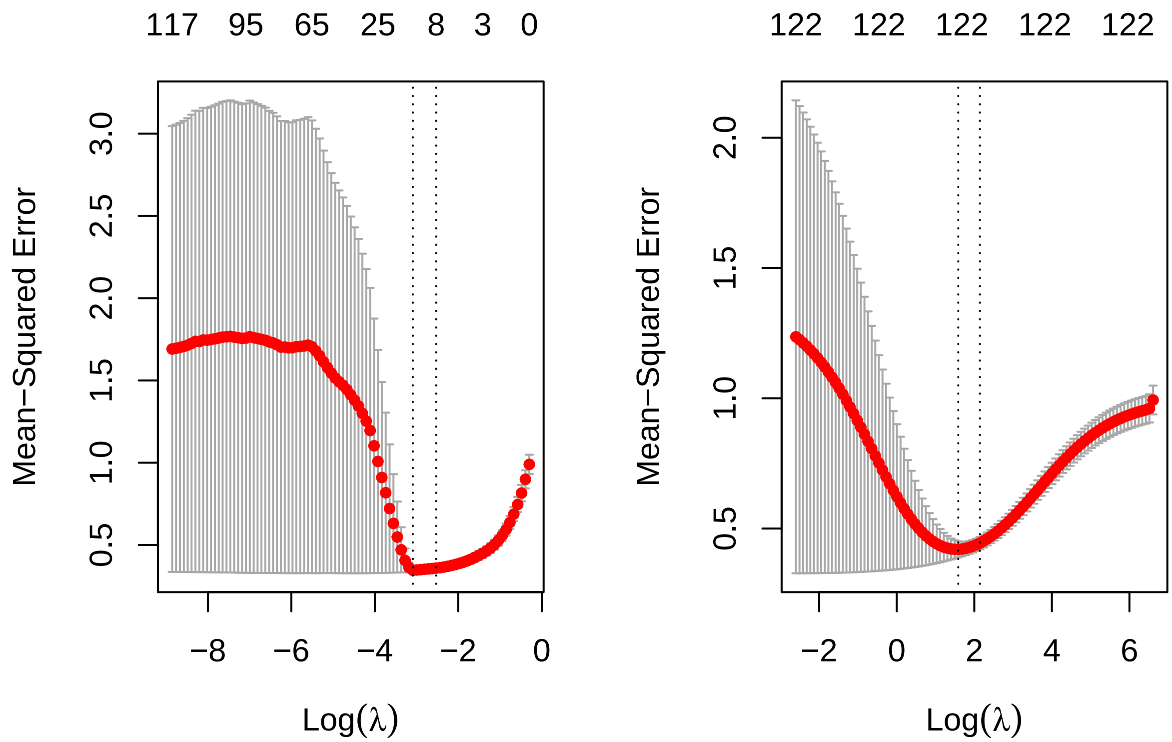
We can see the coefficients in the ridge regression all steadily decrease to 0, where with the LASSO regression, the variable paths are more 'wiggly' and will increase when other variables are set to 0.

Cross Validation

We use the `cv.glmnet` function to perform cross validation on λ. We set the measure to be `"mse"` and stay with the default number of folds which is 10. We then produce plots of the $\log(\lambda)$ values, again with the LASSO on the left and ridge regression on the right:

```
lasso_cv <- cv.glmnet(X_train, y_train, type.measure = "mse", alpha =1)
ridge_cv <- cv.glmnet(X_train, y_train, type.measure = "mse", alpha =0)

par(mfrow = c(1,2))
plot(lasso_cv)
plot(ridge_cv)
```



Here the lines show the $\log(\lambda)$ which minimises the MSE and the largest value of $\log(\lambda)$ such that the corresponding MSE is within one standard error of the minimum.

For this we choose the λs producing the minimum MSE, which are:

```
print(paste("LASSO:", lasso_cv$lambda.min))
```

```
## [1] "LASSO: 0.0455502043376642"
```

```
print(paste("Ridge:", ridge_cv$lambda.min))
```

```
## [1] "Ridge: 4.88419910671498"
```

Comparison of results

We first compare the coefficients resulting from our cross-validations:

```
summary(coef(lasso_cv))
```

```
## 123 x 1 sparse Matrix of class "dgCMatrix", with 9 entries
##    i j             x
## 1  1 1 -0.0005964687
## 2  5 1 -0.2057830097
## 3 40 1  0.0533908716
```

```
## 4 42 1  0.0038587431
## 5 46 1 -0.2636269534
## 6 52 1  0.1828625478
## 7 70 1  0.0455770774
## 8 73 1  0.0723462149
## 9 92 1  0.0450718488
```

```
summary(coef(ridge_cv))
```

```
## 123 x 1 sparse Matrix of class "dgCMatrix", with 123 entries
##       i j             x
## 1     1 1 -4.665502e-03
## 2     2 1  1.097762e-02
## 3     3 1 -1.724367e-03
## 4     4 1  3.226239e-02
## 5     5 1 -3.037919e-02
## 6     6 1  1.199880e-03
## 7     7 1  7.354398e-03
## 8     8 1 -3.713125e-03
## 9     9 1 -9.969908e-04
## 10   10 1 -2.410645e-03
## 11   11 1  2.723730e-03
## 12   12 1  1.145061e-02
## 13   13 1  9.365107e-03
## 14   14 1 -8.327985e-03
## 15   15 1 -8.551916e-03
## 16   16 1 -8.002734e-03
## 17   17 1 -1.880041e-02
## 18   18 1  2.937739e-03
## 19   19 1  1.732528e-02
## 20   20 1 -4.053631e-03
## 21   21 1 -9.384827e-03
## 22   22 1 -5.266277e-03
## 23   23 1  1.478325e-03
## 24   24 1 -6.557919e-03
## 25   25 1 -4.793018e-04
## 26   26 1  3.021269e-04
## 27   27 1 -2.034591e-03
## 28   28 1  5.772151e-05
## 29   29 1  1.236608e-02
## 30   30 1  1.286359e-02
## 31   31 1  9.066156e-03
## 32   32 1  1.319794e-02
## 33   33 1 -7.054643e-03
## 34   34 1  1.209547e-02
## 35   35 1 -7.461867e-03
## 36   36 1 -6.267937e-03
## 37   37 1 -3.861715e-03
## 38   38 1  3.810360e-03
## 39   39 1 -6.878622e-03
## 40   40 1  2.102164e-02
## 41   41 1  9.483465e-03
## 42   42 1  2.117715e-02
## 43   43 1  2.144981e-02
```

```
## 44   44 1  4.599732e-03
## 45   45 1 -2.691873e-02
## 46   46 1 -2.902845e-02
## 47   47 1 -2.471635e-02
## 48   48 1 -2.680581e-02
## 49   49 1 -9.255791e-04
## 50   50 1 -5.914669e-03
## 51   51 1  1.563141e-02
## 52   52 1  3.217789e-02
## 53   53 1  6.730869e-03
## 54   54 1  3.541310e-03
## 55   55 1  4.157258e-03
## 56   56 1  5.817128e-03
## 57   57 1  7.856269e-03
## 58   58 1  6.002165e-03
## 59   59 1  6.740407e-03
## 60   60 1  7.482298e-03
## 61   61 1  8.189002e-03
## 62   62 1 -5.818047e-03
## 63   63 1  6.710048e-03
## 64   64 1  1.257180e-02
## 65   65 1  9.445274e-03
## 66   66 1 -5.330157e-04
## 67   67 1 -4.394542e-03
## 68   68 1  7.774767e-03
## 69   69 1 -1.532677e-02
## 70   70 1  1.440931e-02
## 71   71 1  1.375085e-02
## 72   72 1 -9.084995e-03
## 73   73 1  1.554929e-02
## 74   74 1 -1.362537e-02
## 75   75 1 -1.285207e-02
## 76   76 1  1.905128e-02
## 77   77 1 -6.051694e-04
## 78   78 1  2.083504e-04
## 79   79 1  1.461343e-02
## 80   80 1  1.011105e-02
## 81   81 1 -2.930713e-03
## 82   82 1 -1.730061e-03
## 83   83 1 -6.587337e-04
## 84   84 1 -4.579885e-03
## 85   85 1 -2.728560e-03
## 86   86 1 -2.069518e-03
## 87   87 1 -1.893974e-03
## 88   88 1  1.078908e-02
## 89   89 1  6.552006e-03
## 90   90 1 -3.983191e-04
## 91   91 1  1.222652e-02
## 92   92 1  1.447437e-02
## 93   93 1  6.869012e-03
## 94   94 1 -5.616945e-03
## 95   95 1 -3.112732e-03
## 96   96 1  3.846757e-03
## 97   97 1 -1.088403e-03
```

```
## 98   98 1  1.697002e-03
## 99   99 1 -2.916569e-03
## 100 100 1 -7.868916e-04
## 101 101 1 -2.542820e-03
## 102 102 1  5.136841e-03
## 103 103 1  1.581455e-03
## 104 104 1  6.387748e-03
## 105 105 1 -2.917573e-03
## 106 106 1 -1.510554e-02
## 107 107 1 -1.428430e-02
## 108 108 1  1.962641e-02
## 109 109 1  4.671136e-03
## 110 110 1  3.654369e-03
## 111 111 1  1.605888e-02
## 112 112 1  4.038269e-03
## 113 113 1  2.662755e-03
## 114 114 1  4.408756e-03
## 115 115 1  8.134426e-03
## 116 116 1  8.929409e-03
## 117 117 1  7.916969e-03
## 118 118 1  8.558489e-03
## 119 119 1  3.073143e-03
## 120 120 1 -4.399602e-03
## 121 121 1  5.519635e-05
## 122 122 1  1.435120e-02
## 123 123 1 -2.893454e-03
```

We can see that there are only 9 non-zero entries in our LASSO model, but all the variables are included in the ridge model.

We now use the out-of-sample error obtained from the `predict()` function to compare the ridge regression and LASSO regression we obtained.

```
lasso_residuals <- y_test - predict(lasso_cv, X_test)
ridge_residuals <- y_test - predict(ridge_cv, X_test)

c(sum(lasso_residuals^2), sum(ridge_residuals^2))
```

```
## [1] 207.9464 246.7404
```

We can see the LASSO model has the lower out-of sample prediction error in this case.

Task 2

For this task we use the Bone Mineral Density dataset obtained from https://hastie.su.domains/ElemStatLearn/:

```
data <- read.csv("spnbmd.csv", sep = "\t")
head(data)
```

```
##   idnum   age gender      spnbmd
```

```
## 1      1 11.70   male 0.018080670
## 2      1 12.70   male 0.060109290
## 3      1 13.75   male 0.005857545
## 4      2 13.25   male 0.010263930
## 5      2 14.30   male 0.210526300
## 6      2 15.30   male 0.040843210
```

For this we consider the model

$$Y_i^0 = f(x_i^0) + \varepsilon_i \text{ where } f \in \mathcal{C}^2(\mathbb{R}) \tag{1}$$

where Y_i^0 corresponds the the `spnbmd` variable and x_i^0 corresponds to the `age` column. Further we fit seperate `f` functions for male and female entries.

We first separate the data by gender

```
data_male <- data[data$gender == "male",]
data_female <- data[data$gender == "female",]
```

We then use the `gam()` function to model `spnbmd` with a cubic spline on `age` for both `data_male` and `data_female`. This uses general cross-validation to choose model parameters by default.

```
library(mgcv)
fit_male <- gam(spnbmd ~ s(age, bs = "cr", k =12), data = data_male)
fit_female <- gam(spnbmd ~ s(age, bs = "cc"), data = data_female)
```

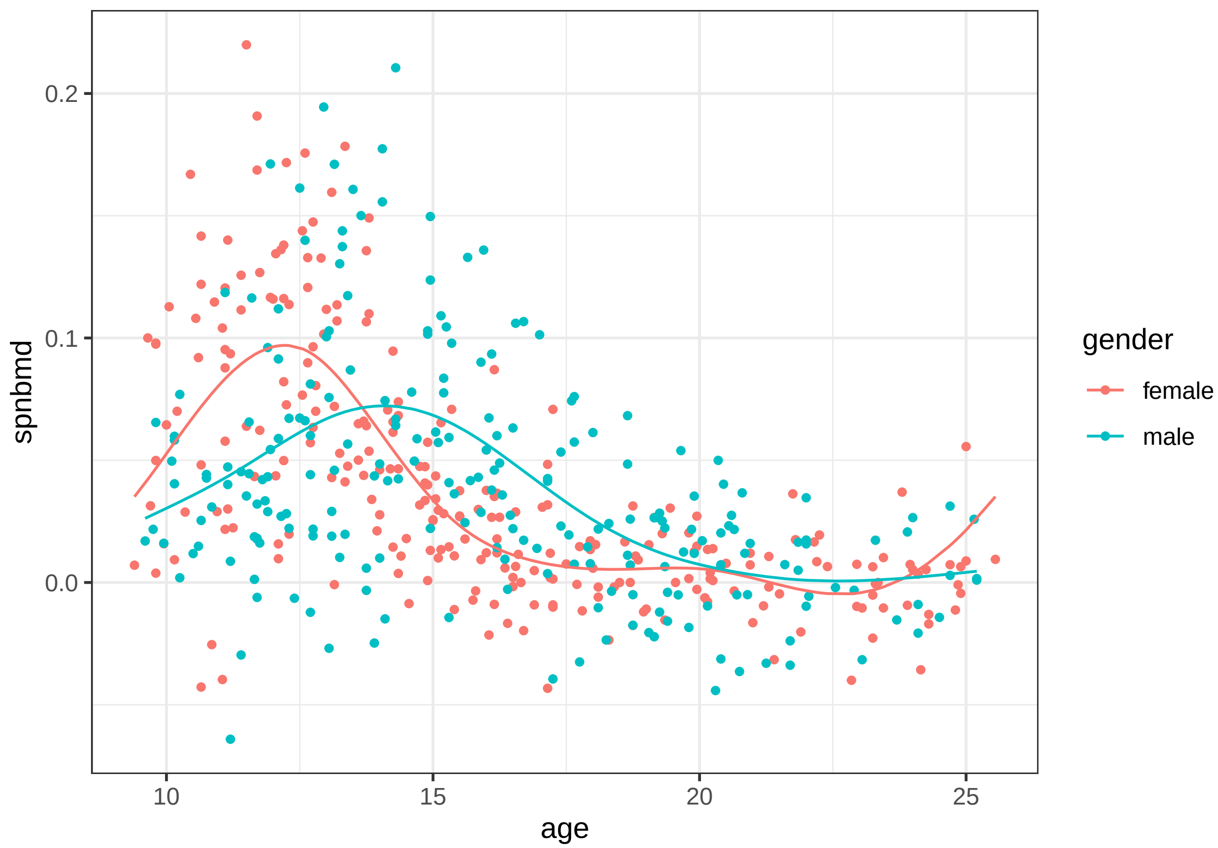
We can plot the estimated functions and data point, colour-coded by gender:

```
library(dplyr)
library(ggplot2)
data_female <- data_female %>%
  as_tibble() %>%
  mutate(fitted = fit_female$fitted.values)

data_male <- data_male %>%
  as_tibble() %>%
  mutate(fitted = fit_male$fitted.values)

data <- rbind(data_female, data_male)

ggplot(data = data, aes(age, spnbmd, color = gender)) +
  geom_point(size = 1) +
  geom_line(aes(y = fitted))
```



0.2
0.1
0.0
spnbmd
10
15
20
25
age
gender
female
male