Computer lab 7

Måns Magnusson

September 12, 2016

Instructions

- This lab should be conducted by students **two by two**.
- The lab constists of writing a package that is version controlled on github.com.
- Both student should **contribute equally much** to the package.
- Commit continously your addition and changes.
- Collaborations should be done using github (ie you should commit using your own github account).
- In the lab some functions can be marked with an *. These parts is only mandatory for students taking the advanced course or students working together in groups of three.
- The deadline for the lab can be found on the **webpage**
- The lab should be turned in as a url to the repository containing the package on github using **LISAM**. This should also include name, github user names and liuid of the students behind the project.

Contents

Chapter 1

Introduction to machine learning in R

In this lab we will continue to improve our `linreg` package by including a similar function for ridge regression.

Master students should implement one of the exercises marked with (*).

1.1 Ridge regression

You should add a new function in your `linreg` package that you call `ridgereg(formula, data, lambda)`. As with the `linreg()` function it should take a formula object as well as a dataset and return a `ridgereg` object. The `ridgereg()` function should also have the argument `lambda` to specify λ.

Ridge regression can be a good alternative when we have a lot of covariates (when $p > n$) or in the situation of multicollinearity. More information on ridge regression can be found in chapter 3.4.1 in [1].

The hyperparameter that we will tune to find the best model is the λ parameters. Unlike the linear regression situation, the different scaling of the covariates in $\mathbf{X}$ will affect the results. So normalize all covariates before you do the analysis.

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sqrt{V(\mathbf{x})}}$$

If you want to compare the results you can compare with `lm.ridge()` in the `MASS` package that is parametrized in the same way. But it uses SVD decomposition so there can be small differences in the results.

1.1.1 Computations using least squares

The simple way to calculate the different is to use ordinary linear algebra and calculate:

Regressions coefficients:

$$\hat{\beta}^{\mathbf{ridge}} = \left(\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

The fitted values:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}^{\mathbf{ridge}}$$

Calculate these statistics and store it in an object of class `ridgereg`. You can use either S3 objects or RC objects.

Document your function `ridgereg()` using `roxygen2`.

1.1.2 (*) Ridge regression using the QR decomposition

As in the situation with linear regression we can do the calculations using the QR decomposition. It is a little bit trickier that in the linear situation, some hints can be found **here**.

1.1.3 Implementing methods

As with the `linreg()` function you should implement some methods for your object.

The following methods should be implemented and be documented using `roxygen2`.

`print()` should **print out** the coefficients and coefficient names, similar as done by the `lm` class.

```
data(iris)
mod_object <- lm(Petal.Length~Species, data = iris)
print(mod_object)


Call:
lm(formula = Petal.Length ~ Species, data = iris)

Coefficients:
      (Intercept)  Speciesversicolor   Speciesvirginica
             1.46               2.80               4.09
```

`predict()` should return the predicted values $\hat{\mathbf{y}}$, it should be able to predict for new dataset similar to the `predict()` function for the `lm()` package.

`coef()` should return the ridge regression coefficients $\hat{\beta}^{\text{ridge}}$

1.1.4 Write a test suite

Write a simple test suite for your `ridgereg()` and associated methods.

1.1.5 Handling large datasets with `dplyr`

Create a function you call `visualize_airport_delays()` without any arguments that creates a plot that visualizes the mean delay of flights for different airports by longitude and latitude using `ggplot2`. The datasets can be found in the `nycflights13` package.

The data handling should be done using `dplyr` verbs as much as possible. See the cheat sheet **here** for more information how to data munge using `dplyr`. Remember that `delays` is a variable in the `flights` dataset and airport information is in the `airports` dataset.

1.2 Create a vignettes for `ridgereg()`, `dplyr` and the `caret` package

Create a vignette called `ridgereg` where you show how to do a simple prediction problem using your own `ridgereg()` function.

Use the caret package and your `ridgereg()` function to create a predictive model for the `BostonHousing` data found in the `mlbench` package or (*) data from your own API.

The vignette should include the following:

1. Divide the `BostonHousing` data (or your own API data) into a test and training dataset using the `caret` package.

2. Fit a linear regression model and a fit a linear regression model with forward selection of covariates on the training dataset. Information on linear regression models in the `caret` package can be found **here**.

3. Evaluate the performance of this model on the training dataset.

4. Fit a ridge regression model using your `ridgereg()` function to the training dataset for different values of λ. How to include custom models in caret is described **here**.

5. Find the best hyperparameter value for λ using 10-fold cross-validation on the training set. More information how to use the `caret` package for training can be found **here** and **here**.

6. Evaluate the performance of all three models on the test dataset and write some concluding comments.

1.2.1 (*) Predictive modeling of flight delays using `ridgereg()`

Create a new vignette called flight_delay where you try to predict the delay of each flight using your own `ridgereg()` function. If the data is to large, you can scale it down a bit, but the purpose is to try to do predictions using larger datasets.

1. Read in the weather dataset and the flights dataset from the `nycflights13` package and remove eventual variables you do not believe to have a predictive value.

2. Add extra weather data from the weather dataset and create interaction effects you think can be of interest for the prediction.

3. Use the caret package to divide the flight dataset into three sets: test, train and validation (with the proportions 5%, 80% and 15%.

4. Train ridge regressions models for different values of λ and evaluate the root mean squared error (see **here**) on the validation set. Try to find a optimal value for λ.

5. When you found a good value for λ, use this to predict the test set and report the RMSE of your predicted model.

1.3 Seminar and examination

During the seminar you will bring your own computer and demonstrate your package and what you found difficult in the project.

We will present as many packages as possible during the seminar and you should

1. Show that the package can be built using R Studio and that all unit tests is passing.

2. Show your vignette/analysis.

1.3.1 Examination

Turn in a the adress to your github repo with the package using LISAM.

The packages will be assigned to other groups to try your package out and return eventual problems as bugs using the issue tracker. The teacher will the decide if there are any bugs or corrections that is needed to correct to get the lab to pass.

Bibliography

[1] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.