Lab 1a - Regularized regression

Machine Learning 7.5 credits

Mattias Villani, Department of Statistics, Stockholm University

INSTRUCTIONS:

- The sections named Intro do **not** have any problems for you. Those sections contain code used to set up the data and do some initial analysis, so just read and follow along by running each code chunk.
- Your problems are clearly marked out as Problem 1, Problem 2 etc. You should answer all problems by adding code chunks and text below each question.
- Your submission in Athena should contain two files:
  - This Rmd file with your answers.
  - A PDF version of this file (use the knit to PDF option above).
- You can also write math expression via LaTeX, using the dollar sign, for example β.
- You can navigate the sections of this file clicking (Top Level) in the bottom of RStudio's code window.

Intro1 - Loading packages and data

Do `install.packages()` for each package the first time you use a new package.

```
suppressMessages(library(dplyr)) # Data transformations and tables
library("RColorBrewer") # for pretty colors
colors = brewer.pal(12, "Paired")[c(1,2,7,8,3,4,5,6,9,10)]
set.seed(123342)         # set the seed for reproducability
```

The dataset we will work with here contains the number of rides by a bike sharing company. This dataset is hourly observations over the two-year period January 1, 2011 - December 31, 2012. The total number of observations is 17379. The dataset contains many covariates that may be useful for predicting the number of rides: the hour of the day, temperature, humidity, season, weekday etc.

In the first part of the lab you will concentrate on modeling the number of rides as function of the time of the day. Since the number of rides (`cnt`) is a non-negative count variable we will use the logarithm (`logrides`) as the response variable. Note that the smallest value of `cnt` is 1, so we will not have any problems with log(0).
Let's load the data, make some transformations and have look at the raw data.

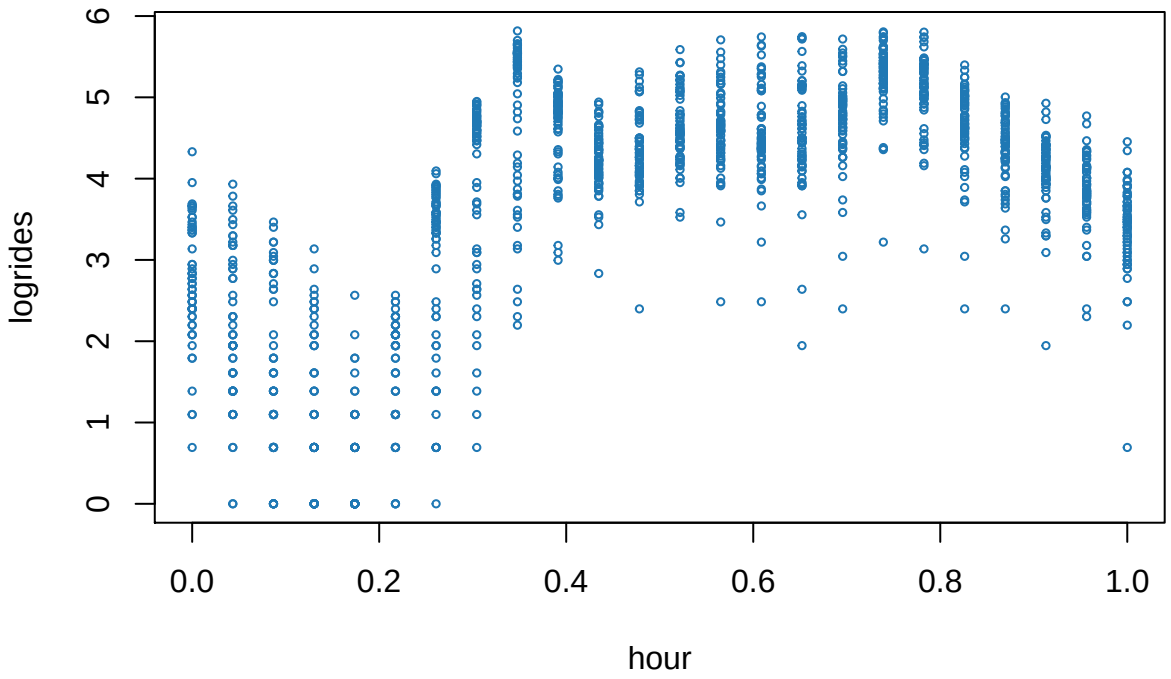
```
bikes = read.csv("https://github.com/mattiasvillani/MLcourse/raw/main/Data/BikeShareData/hour.csv")
bikes$dteday = as.Date(bikes$dteday) # convert date column to proper date format
bikes$logrides = log(bikes$cnt)      # we model the log(number of rides) as response.
bikes$hour = bikes$hr/23             # hour of the day. midnight is 0, 11 PM is 1.
head(bikes)
```

```
##   instant     dteday season yr mnth hr holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1  0       0       6          0          1
## 2       2 2011-01-01      1  0    1  1       0       6          0          1
## 3       3 2011-01-01      1  0    1  2       0       6          0          1
## 4       4 2011-01-01      1  0    1  3       0       6          0          1
## 5       5 2011-01-01      1  0    1  4       0       6          0          1
```

```
## 6       6 2011-01-01      1  0    1  5       0       6          0          2
##   temp  atemp  hum windspeed casual registered cnt logrides       hour
## 1 0.24 0.2879 0.81    0.0000      3         13  16 2.772589 0.00000000
## 2 0.22 0.2727 0.80    0.0000      8         32  40 3.688879 0.04347826
## 3 0.22 0.2727 0.80    0.0000      5         27  32 3.465736 0.08695652
## 4 0.24 0.2879 0.75    0.0000      3         10  13 2.564949 0.13043478
## 5 0.24 0.2879 0.75    0.0000      0          1   1 0.000000 0.17391304
## 6 0.24 0.2576 0.75    0.0896      0          1   1 0.000000 0.21739130
```

We will start by training on only 2 months, Febrary and March of 2011, and only using the variable **hour** as predictor for **logrides**. As seen in the figure below, the expected the number of rides over the day seems to peak in the morning (8 AM is $8/23 \approx 0.35$) and after work (6 PM is $18/23 \approx 0.78$).

```
bikesTrain = bikes[bikes$dteday >= as.Date("2011-02-01") &
                     bikes$dteday <= as.Date("2011-03-31"),] # Data from feb and march 2011
plot(bikesTrain$hour, bikesTrain$logrides, xlab = "hour", ylab = "logrides",
     col = colors[2], cex = 0.5)
```



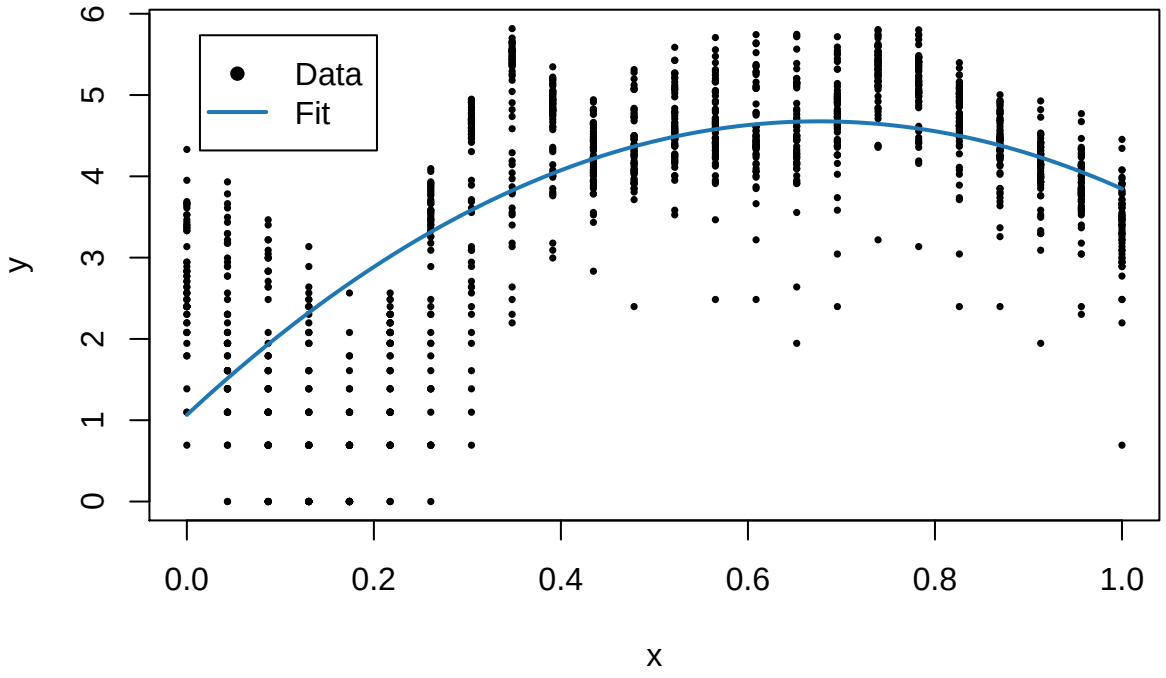
We will now fit a quadratic regression of **logRides** against the covariate **hour** to the training data (Feb 1, 2011 - March 31, 2011) and compute the training RMSE. We can then use the trained model to predict **logRides** on the test set consisting of the following 2 months between April 1, 2011 - May 31, 2011.

```
# Function that computes the basis function for a vector of x-values.
# order=2 (quadratic) in the example above.
PolyMatrix <- function(x, order){
    X = cbind(1,x)
    if (order==1){return(X)}
    for (k in 2:order){
        X = cbind(X, x^k)
    }
    return(X)
}

# Training a quadratic model
```

```
order = 2
XTrain = PolyMatrix(bikesTrain$hour, order)
yTrain = bikesTrain$logrides
betaHat = solve(crossprod(XTrain),crossprod(XTrain,yTrain)) # Least squares estimator
yFit = XTrain%*%betaHat

# Function that trains a polynomial model,
# computes predictions over fine grid of values xGrid
# and plots the fit and training data.
PolyPlotFit <- function(x, y, order, xGrid){
    X = PolyMatrix(x, order)
    betaHat = solve(crossprod(X),crossprod(X,y))
    Xgrid = PolyMatrix(xGrid, order)
    yFit = Xgrid%*%betaHat
    plot(x, y, pch = 16, cex = 0.5)
    lines(xGrid, yFit, col = colors[2], lwd = 2)
    legend(x = "topleft", inset=.05, legend = c("Data", "Fit"),
       lty = c(NA, 1), lwd = c(2, 2), pch = c(16, NA),
       col = c("black", colors[2]))
}
xGrid = seq(0, 1, length = 100)
PolyPlotFit(bikesTrain$hour, bikesTrain$logrides, order, xGrid)
```




```
# Prediction on test data in April-May
bikesTest = bikes[bikes$dteday >= as.Date("2011-04-01") &
                  bikes$dteday <= as.Date("2011-05-31"),] # Test on following two months
XTest = PolyMatrix(bikesTest$hour, order)
yTest = bikesTest$logrides
yPred = XTest%*%betaHat

RMSEtrain = sqrt(sum((yTrain-yFit)^2)/length(yTrain))
RMSEtest = sqrt(sum((yTest-yPred)^2)/length(yTest))
```

```
message(paste("Training RMSE: ",RMSEtrain))
```

```
## Training RMSE:  0.987230880106032
```

```
message(paste("Test RMSE: ",RMSEtest))
```

```
## Test RMSE:  1.20881698667509
```

Problem 1 - Polynomial regression Code everything from scratch on this problem, no `lm()` or anything. You **can** however use the functions defined in this note book, for example `PolyPlotFit`. Do this:

- Fit a polynomial regression of `logRides` against the covariate `hour` to the training data (Feb 1, 2011 - March 31, 2011) using a polynomials of order 8. Plot the fit of the model overlayed on the scatter of training data.
- Fit polynomials with order varying between 1 and 10 on the training data. Plot the training RMSE as a function of the polynomial order.
- For each polynomial order between 1 and 10, use the trained model to predict the `logRides` on the test set consisting of the following 2 months between April 1, 2011 - May 31, 2011. Compute the test RMSE for each polynomial order and plot it in the same plot as the training RMSE.
- Comment on the difference of the RMSE on the training and test data: are we overfitting or underfitting the data? Other explanation of the results?

Problem 2 - Spline regression with L2 regularization Use the package `glmnet` to fit a spline regression with equally spaced knots for `hour` between 0.05 and 0.95. Use L2-regularization and find the optimal λ by 10-fold cross-validation on the training data using the one-standard deviation rule (lambda.1se). Use the `splines` package in R to create natural cubic splines basis functions with 10 degrees of freedom, i.e. use the `ns()` function with df=10 as input argument. Compute the RMSE in training (Feb-March) and test (April-May).

Problem 3 - Spline regression with L1 regularization Repeat Problem 2, this time using L1 regularization.

Intro2 - using all data and all covariates Let us now get a little more serious and use many more of the covariates and all of the data. **The first year and a half of data will be used for training and the remaining half year for testing**. But let's first turn the categorical covariates in the dataset into dummy variables (one-hot encoded variables). We will follow the statistical approach of using $K-1$ dummy variables for a categorical variable with K levels. The first level will be the reference category. This little function will work for the current datasets where all levels are numerical:

```
onehot <- function(x){
    levels = sort(unique(x))
    onehotMatrix = matrix(0, length(x), length(levels)-1)
    count = 0
    for (level in levels[-1]){
        count = count + 1
        onehotMatrix[x == level, count] = 1
    }
    return(onehotMatrix)
}
```

[Note: an alternative approach to constructing one-hot dummy variables is to use R's concept of `factor` variables. You can do the one-hot variables from the factor variable construction and the `model.matrix` function.]

Let's construct the one-hot dummies for: - the `weathersit` variable (clear weather is the reference category) - the `weekday` variable (0 = Sunday is the reference category) - the `season` variable (spring is the reference)

```
weatherOneHot = data.frame(onehot(bikes$weathersit))
names(weatherOneHot) <- c("cloudy", "lightrain","heavyrain")
bikes = cbind(bikes, weatherOneHot)

weekdayOneHot = data.frame(onehot(bikes$weekday))
names(weekdayOneHot) <- c("mon","tue","wed","thu","fri","sat")
bikes = cbind(bikes, weekdayOneHot)

seasonOneHot = data.frame(onehot(bikes$season))
names(seasonOneHot) <- c("summer", "fall","winter")
bikes = cbind(bikes, seasonOneHot)

head(bikes) # always look at the to see that we didn't mess it up.
```

```
##   instant     dteday season yr mnth hr holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1  0       0       6          0          1
## 2       2 2011-01-01      1  0    1  1       0       6          0          1
## 3       3 2011-01-01      1  0    1  2       0       6          0          1
## 4       4 2011-01-01      1  0    1  3       0       6          0          1
## 5       5 2011-01-01      1  0    1  4       0       6          0          1
## 6       6 2011-01-01      1  0    1  5       0       6          0          2
##   temp  atemp  hum windspeed casual registered cnt logrides       hour cloudy
## 1 0.24 0.2879 0.81    0.0000      3         13  16 2.772589 0.00000000      0
## 2 0.22 0.2727 0.80    0.0000      8         32  40 3.688879 0.04347826      0
## 3 0.22 0.2727 0.80    0.0000      5         27  32 3.465736 0.08695652      0
## 4 0.24 0.2879 0.75    0.0000      3         10  13 2.564949 0.13043478      0
## 5 0.24 0.2879 0.75    0.0000      0          1   1 0.000000 0.17391304      0
## 6 0.24 0.2576 0.75    0.0896      0          1   1 0.000000 0.21739130      1
##   lightrain heavyrain mon tue wed thu fri sat summer fall winter
## 1         0         0   0   0   0   0   0   1      0    0      0
## 2         0         0   0   0   0   0   0   1      0    0      0
## 3         0         0   0   0   0   0   0   1      0    0      0
## 4         0         0   0   0   0   0   0   1      0    0      0
## 5         0         0   0   0   0   0   0   1      0    0      0
## 6         0         0   0   0   0   0   0   1      0    0      0
```

Problem 4 - Spline regression with L1 regularization with more covariates Use `glmnet` to estimate an L1 regularized regression for the following regression model expressed for clarity as an R formula:

logrides ~ s(hour) + yr + holiday + workingday + temp + atemp + hum + windspeed + weekdayDummies + seasonDummies + weatherDummies,

where

- s(hour) are spline terms, so that s(hour) means adding all the splines covariates to the model, one for each knot (in addition to the linear term)
- weekdayDummies, seasonDummies and weatherDummies, each means to add all the one-hot covariates for each these three effects.

Compute the RMSE on the training (Jan 1, 2011 - May 31, 2012) and the test data (June 1, 2012- Dec 31, 2012). Which three covariates seem to be most important in the training data?

Problem 5 - Time series effects in a regression So far we have ignored that the data is a time series. Let us now check if the residuals are autocorrelated, and then try to improve on the model by adding time

series effects in the regression.
Do the following steps:

a) Plot the autocorrelation function for the residuals in the training data from the previously fitted L1-regularized regression (hint: acf()). Comment.

b) Plot the actual time series for the last 24*7 observations in the dataset (the last week) and the prediction for those values in the same graph. Here you should plot the data on the original scale, i.e. plot exp(logrides) and exp() of the predictions.

c) Add time series effects by adding the first four hourly lags and the 24th hourly lag to the set of covariates. (hint: lag() and note that you loose observations when taking lags). Fit the L1-regularized regression with all previous covariates and the new time lags. Compute RMSE in training and in test.

d) Plot the actual time series for the last $24 * 7 = 168$ observations in the dataset (the last week) and the prediction for those values in the same graph.

Problem 6 - Regression trees We will now fit a regression tree.

a) Use the training dataset created in problem 5 c, this time to fit regression trees from the `tree` package. Compute RMSE for the training and test sets. Use the default settings when reporting the results, but feel free to experiment with the settings to see the effect of changing them. Plot the fitted model to show the tree structure.

b) Plot the actual time series for the last 24∗7=168 observations in the dataset (the last week) along with the predictions from the tree model.

Problem 7 - Random forest Repeat Problem 6, this time using the random forest regression model from the `randomForest` package. No need to plot the trees or the forest, however.

The default setting of the randomForest function is to train 500 trees. You can speed up the function by specifying a lower number of trees with the argument ntree.

The argument sampsize allows you to train each tree using only a sample of the training set. A smaller sample size speeds up the function.

You can test a few different combinations of ntree and sampsize. Is it better in terms of accuracy to train many trees, each with a smaller sample size, or to train fewer trees each with the full training set?

Problem 8 - XGboost Repeat Problem 6, this time using the xgboost regression model from the `xgboost` package. Use the argument nrounds = 25 when reporting the results, which means that the process iterates 25 times, but feel free to experiment with the settings to see the effect of changing them.