

# Module 7: Recommended Exercises

TMA4268 Statistical Learning V2020

Andreas Strand, Martina Hall, Michail Spitieris, Stefanie Muff, Department of Mathematical Sciences, N

February 27, 2020

## Problem 1

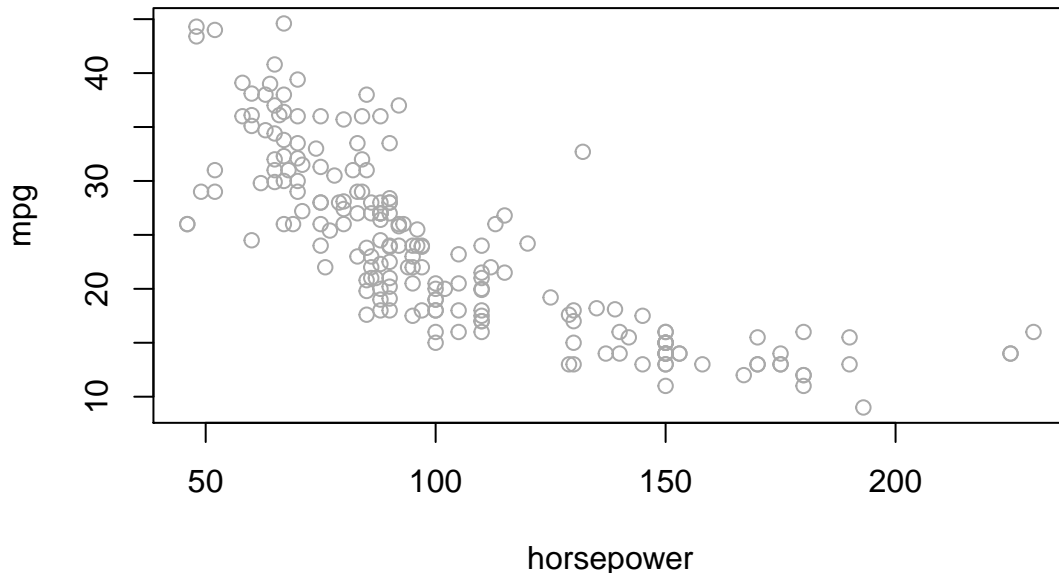
Let us take a look at the `Auto` data set. We want to model miles per gallon `mpg` by engine horsepower `horsepower`. Separate the observations into training and test. A training set is plotted below.

Perform polynomial regression of degree 1, 2, 3 and 4. Use `lines()` to add the fitted values to the plot below.

Also plot the test error depending on polynomial degree.

```
library(ISLR)
# extract only the two variables from Auto
ds = Auto[c("horsepower", "mpg")]
n = nrow(ds)
# which degrees we will look at
deg = 1:4
set.seed(1)
# training ids for training set
tr = sample.int(n, n/2)
# plot of training data
plot(ds[tr, ], col = "darkgrey", main = "Polynomial regression")
```

## Polynomial regression



## Problem 2

We will continue working with the `Auto` data set. The variable `origin` is 1,2 or 3, corresponding to American, European or Japanese origin. Use `factor(origin)` for conversion to a factor variable. Predict `mpg` by origin with a linear model. Plot the fitted values and approximative 95 percent confidence intervals. Selecting `se = T` in `predict()` gives standard errors of the prediction.

- Hint: make a new dataframe of the three origins (as factors) and use this new data in your predict function.
- Hint: to plot the confidence intervals, you can add `geom_segment(aes(x=origin, y=lwr, xend = origin, yend=upr))` to your ggplot, where `origin`, `lwr` and `upr` comes from a dataframe with `lwr` as the lower bound and `upr` as the upper bound.

## Problem 3

Now, let us look at the `Wage` data set. The section on [Additive Models \(slides 26-32 in the pdf\)](#) explains how we can create an AM by adding components together. One component we saw is a natural spline in `year` with one knot. Derive the expression for the design matrix  $\mathbf{X}_2$  from the natural spline basis:

$$b_1(x_i) = x_i, \quad b_{k+2}(x_i) = d_k(x_i) - d_K(x_i), \quad k = 0, \dots, K-1,$$

$$d_k(x_i) = \frac{(x_i - c_k)_+^3 - (x_i - c_{K+1})_+^3}{c_{K+1} - c_k}.$$

## Problem 4

We will continue working with the same AM as in problem 3. The R call `model.matrix(~bs(age,knots=c(40,60)) + ns(year,knots=2006) + education)` gives a design matrix for the AM. This matrix is what `gam()` uses. However, it does not equal our design matrix for AM  $\mathbf{X} = (\mathbf{1}, \mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$ . The predicted responses will still be the same.

Write code that produces  $\mathbf{X}$ . The code below may be useful.

```
# X_1
mybs = function(x, knots) {
  cbind(x, x^2, x^3, sapply(knots, function(y) pmax(0, x - y)^3))
}

d = function(c, cK, x) (pmax(0, x - c)^3 - pmax(0, x - cK)^3)/(cK - c)
# X_2
myns = function(x, knots) {
  kn = c(min(x), knots, max(x))
  K = length(kn)
  sub = d(kn[K - 1], kn[K], x)
  cbind(x, sapply(kn[1:(K - 2)], d, kn[K], x) - sub)
}
# X_3
myfactor = function(x) model.matrix(~x)[, -1]
```

If the code is valid, the predicted response  $\hat{\mathbf{y}} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$  should be the same as when using the built-in R function `gam()`.

**R-hints:**

```

# install.packages('gam')
library(gam)
library(ISLR)
# Figure out how to define your X-matrix (this is perhaps a bit tricky!)
X = ...
# fitted model with our X
myhat = lm(wage ~ X - 1)$fit
# fitted model with gam
yhat = gam(wage ~ bs(age, knots = c(40, 60)) + ns(year, knots = 2006) + education)$fit
# are they equal?
all.equal(myhat, yhat)

```

How can myhat equal yhat when the design matrices differ?

## Problem 5

In this exercise we take a quick look at different non-linear regression methods. We continue using the Auto dataset from above, but with more variables.

Fit an additive model using the function `gam` from package `gam`. Call the result `gamobject`.

- `mpg` is the response,
- `displace` is a cubic spline (hint: `bs`) with one knot at 290,
- `horsepower` is a polynomial of degree 2 (hint: `poly`),
- `weight` is a linear function,
- `acceleration` is a smoothing spline with `df=3` (hint: `s`),
- `origin` is a categorical variable (which can be interpreted as a step function, which can be seen from the dummy variable coding).

Plot the resulting curves. Comment on what you see.

**R-hints:** first set `par(mfrow=c(2,3))` and then `plot(gamobject, se=TRUE, col="blue")`.