Module 7: Recommended Exercises

TMA4268 Statistical Learning V2025

Sara Martino, Stefanie Muff, Kenneth Aase
Department of Mathematical Sciences, NTNU

Feb 28, 2025

The original version of this exercise sheet was developed in 2019 by Andreas Strand and colleagues. Thanks for the permission to build on it.

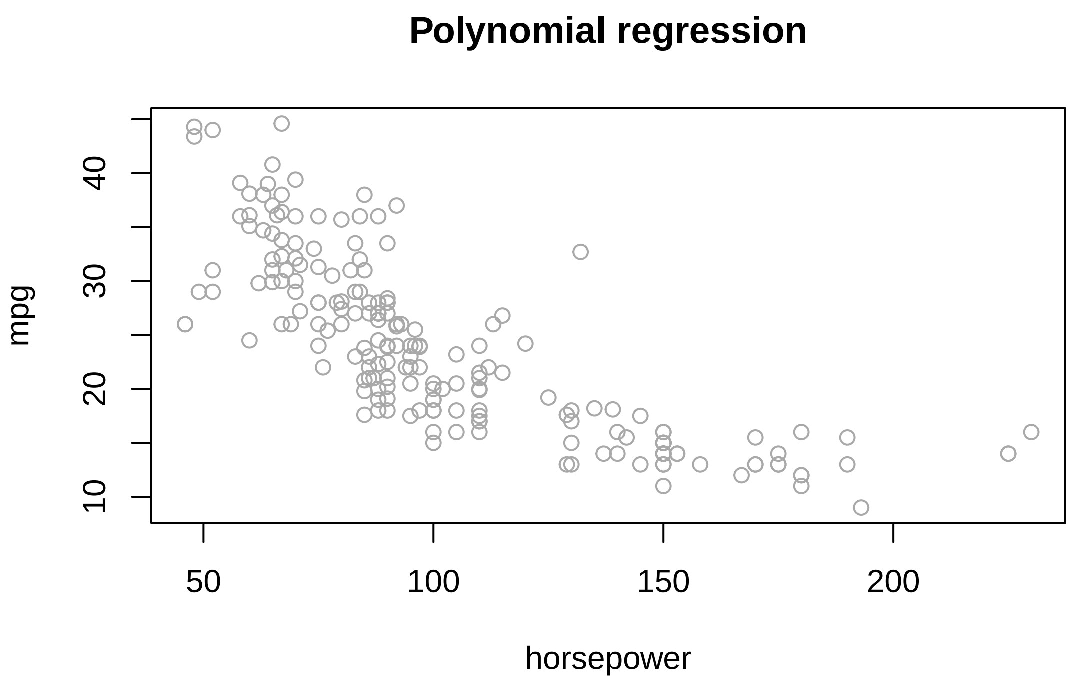
Problem 1

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.

```r
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, round(n / 2))
# plot of training data
plot(ds[tr, ], col = "darkgrey", main = "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, respectively. Use `factor(origin)` for conversion to a factor variable. Predict `mpg` by origin with a linear model. Plot the fitted values and approximative 95% confidence intervals. Selecting `se = TRUE` in `predict()` gives standard errors of the prediction.

- Hint: make a new `data.frame` 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 (in this week's slides) explains how we can create an additive model by adding components together. One type of component we saw is natural cubic splines. Derive the design matrix $\mathbf{X}$ for a natural cubic spline with one internal knot at `year` $= 2006$, from the natural cubic spline basis:

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

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

Problem 4

We will continue working with the same additive model as in problem 3, but we now also include a cubic spline for the covariate `age`, and a linear term for `education`. The `R` call `model.matrix(~ bs(age, knots = c(40, 60)) + ns(year, knots = 2006) + education)` gives a design matrix for the model. This matrix is what gam() uses. However, it does not equal our design matrix for the additive model $\mathbf{X} = (\mathbf{1}, \mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$. Despite this, 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")`).