Solutions to Recommended Exercises 7

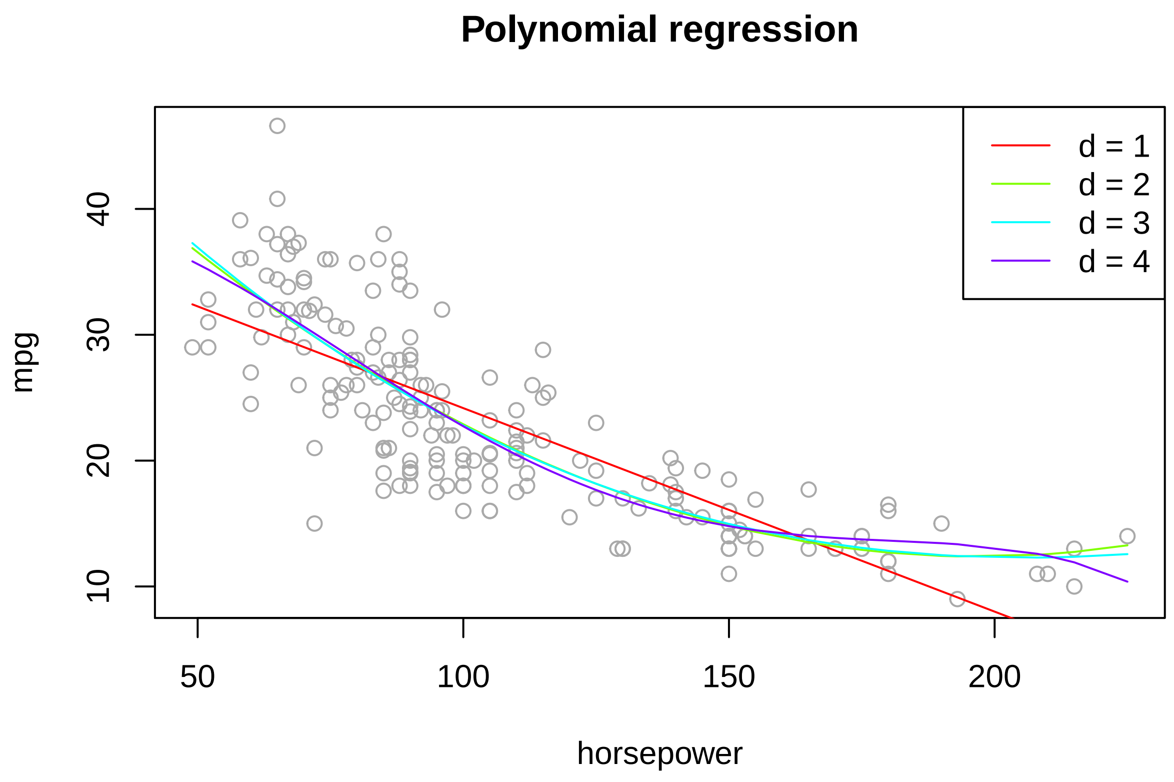
TMA4268 Statistical learning

Andreas Strand

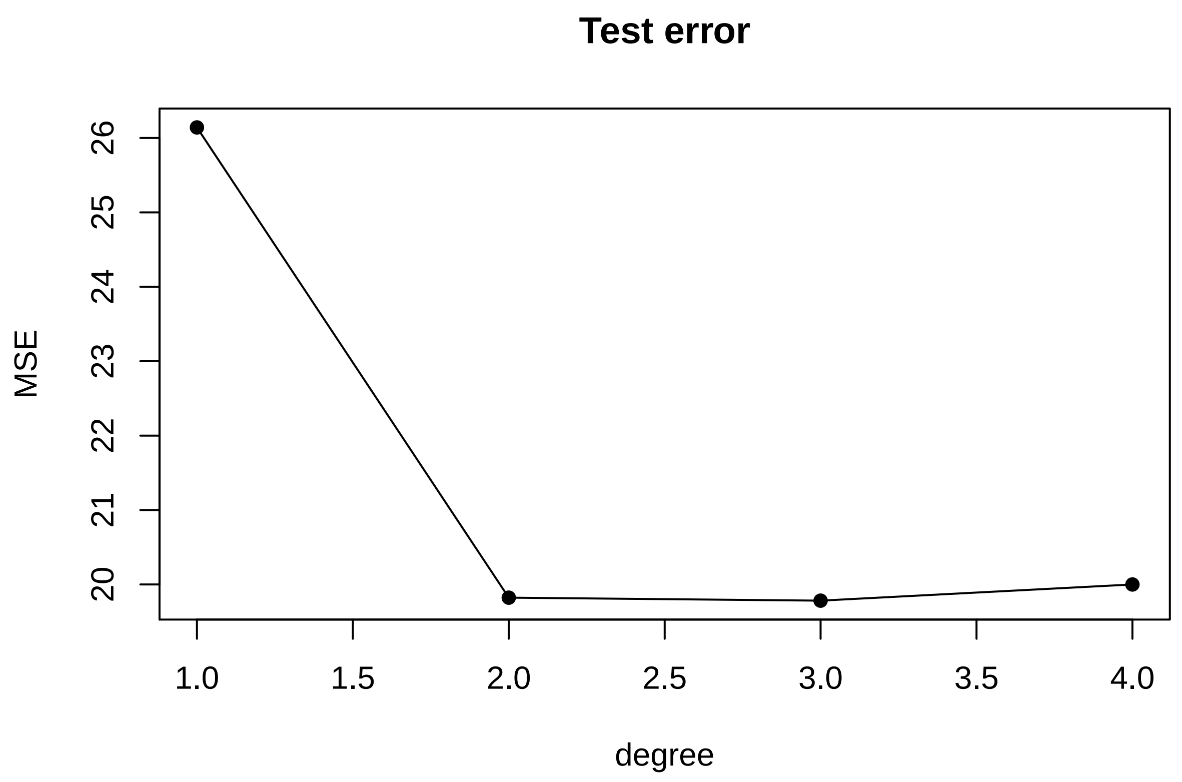
14 mai, 2018

Problem 1: The code below performs polynomial regression of degree 1, 2, 3 and 4. The function `sapply()` is an efficient for loop. We iterate over all degrees to plot the fitted values and compute the test error. Finally we plot the test error by polynomial degree.

```r
library(ISLR)
ds = Auto[c("horsepower","mpg")]
n = nrow(ds)
set.seed(1)
tr = sample.int(n, n/2)
plot(ds[tr,], col = "darkgrey", main = "Polynomial regression")
deg = 1:4
co = rainbow(length(deg))
MSE = sapply(deg, function(d){
  mod = lm(mpg ~ poly(horsepower,d),ds[tr,])
  lines(cbind(ds[tr,1],mod$fit)[order(ds[tr,1]),], col = co[d])
  mean((predict(mod,ds[-tr,])- ds[-tr,2])^2)
})
legend("topright", legend = paste("d =",deg), lty = 1, col = co)
```

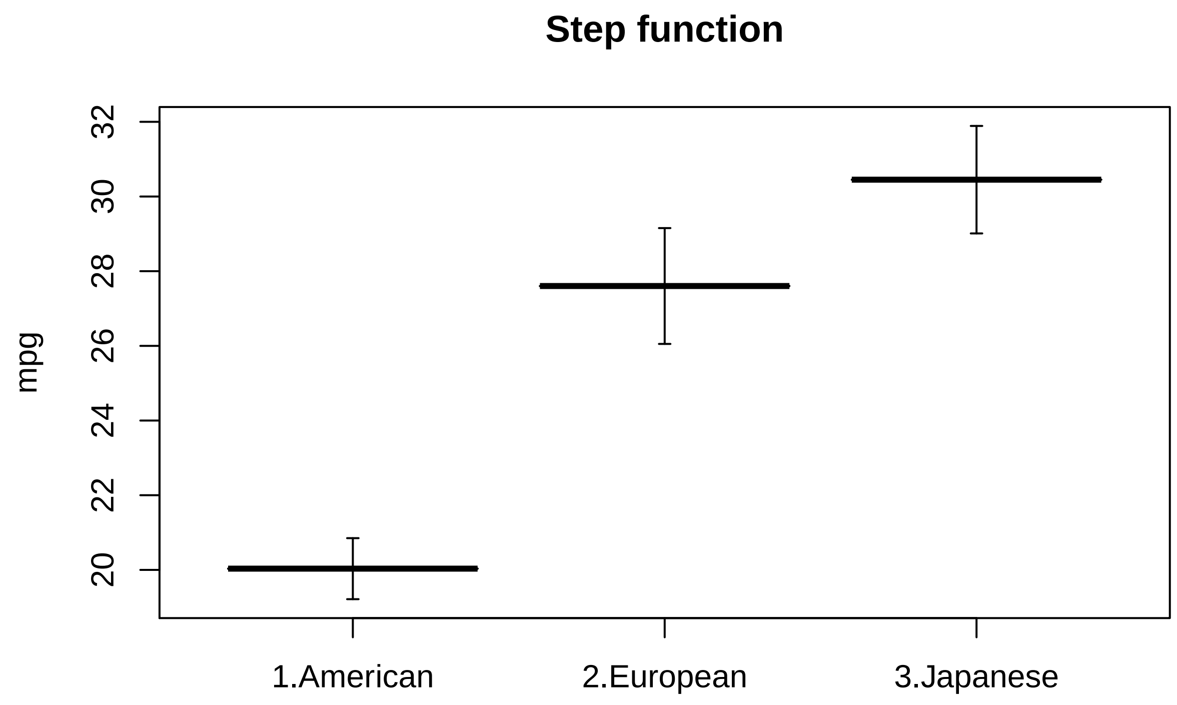


```r
plot(MSE, type="o", pch = 16, xlab = "degree", main = "Test error")
```



Problem 2: We use `factor(origin)` for conversion to a factor variable. The function `predict(..., se = T)` gives fitted values with standard errors.

```r
attach(Auto)
fit = lm(mpg ~ factor(origin))
new = sort(unique(origin))
pred = predict(fit, list("origin" = new), se=T)
se.b = cbind(pred$fit + 2*pred$se.fit, pred$fit- 2*pred$se.fit)
plot(factor(c("1.American", "2.European", "3.Japanese")),
     pred$fit, ylim = range(se.b), main = "Step function",ylab = "mpg")
arrows(new, se.b[,1], y1 = se.b[,2],angle = 90, length = 0.03, code = 3)
```



Problem 3: The request is a design matrix for a natural spline with $X =$ `year` and one knot $c_1 = 2006$. The boundary knots be the extreme values of `year`, that is $c_0 = 2003$ and $c_2 = 2009$. A general basis for a natural spline is

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

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

In our case we have one internal knot, that is $K = 1$. Thus, k takes only the value 0. The two basis functions are

$$\begin{aligned}
b_1(x_i) &= x_i,\\
b_2(x_i) &= d_0(x_i) - d_1(x_i)\\
&= \frac{(x_i - c_0)_+^3 - (x_i - c_2)_+^3}{c_2 - c_0} - \frac{(x_i - c_1)_+^3 - (x_i - c_2)_+^3}{c_2 - c_1}\\
&= \frac{1}{c_2 - c_0}(x_i - c_0)_+^3 - \frac{1}{c_2 - c_1}(x_i - c_1)_+^3 + \left(\frac{1}{c_2 - c_1} - \frac{1}{c_2 - c_0}\right)(x_i - c_2)_+^3\\
&= \frac{1}{6}(x_i - 2003)_+^3 - \frac{1}{3}(x_i - 2006)_+^3 + \frac{1}{6}(x_i - 2009)_+^3.
\end{aligned}$$

The design matrix is obtained by $\{\mathbf{X}_2\}_{ij} = b_j(x_i)$. We can simplify the second basis function more by using the fact that the boundary knots are the extreme values of x_i, that is $2003 \le x_i \le 2009$. Thus,

$$b_2(x_i) = \frac{1}{6}(x_i - 2003)^3 - \frac{1}{3}(x_i - 2006)_+^3.$$

Problem 4: The matrix $\mathbf{X}$ is obtained by using `cbind()` to join an intercept, a cubic spline, a natural cubic spline and a factor.

```r
attach(Wage)
library(gam)
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)
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)
}

myfactor = function(x) model.matrix(~x)[,-1]

X = cbind(1,mybs(age,c(40,60)), myns(year, 2006), myfactor(education))
myhat = lm(wage~X-1)$fit
yhat = gam(wage ~ bs(age, knots = c(40,60)) + ns(year, knots = 2006) + education)$fit
all.equal(myhat,yhat)
```

```
## [1] TRUE
```

The fitted values `myhat` and `yhat` are equal. Both the design matrices $\mathbf{X}$ and the coefficients $\hat{\beta}$ differs, but $\mathbf{X}\hat{\beta}$ are the same.

Problem 5: Solution to problem 3 on Compulsory 2.

Problem 6: The fitted values are obtained by $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$. In R this is `S%*%y`. The effective degrees of freedom is defined as

$$df_\lambda = \sum_{i=1}^{n} \frac{1}{1+\lambda d_i}.$$

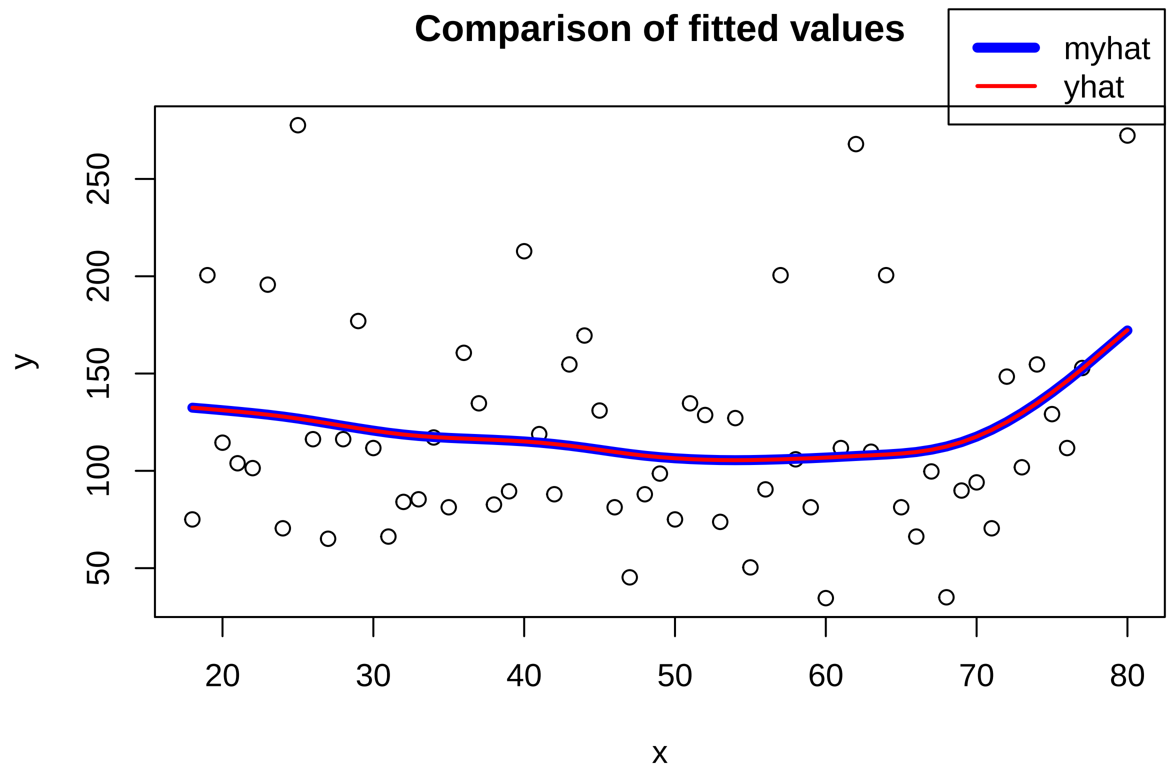
This summation is done in one line in R.

```r
K = function(x){
  xi = sort(unique(x))
  n = length(xi)
  h = xi[-1]-xi[-n]
  i = seq.int(n-2)
  D = diag(1/h[i], ncol = n)
  D[cbind(i,i+1)] = - 1/h[i] - 1/h[i+1]
  D[cbind(i,i+2)] = 1/h[i+1]
  W = diag(h[i]+h[i+1]/3)
  W[cbind(i[-1],i[-1]-1)] = h[i[-1]]/6
  W[cbind(i[-1]-1,i[-1])] = h[i[-1]]/6
  t(D)%*%solve(W)%*%D
}

lambda = 2000
x = sort(unique(age))
y = wage[order(age[!duplicated(age)])]
eig = eigen(K(x))
U = eig$vectors
d = eig$values
S = U%*%diag(1/(1+lambda*d))%*%t(U)
```

```r
myhat = S%*%y
yhat = smooth.spline(x, y, df = sum(1/(1+lambda*d)))$y

plot(x,y, main = "Comparison of fitted values")
co = c("blue", "red")
w = c(5,2)
lines(x,myhat, lwd = w[1], col = co[1])
lines(x,yhat, lwd = w[2], col = co[2])
legend("topright", legend = c("myhat", "yhat"), col = co,
       lwd = w, inset=c(0, -0.19), xpd = T)
```


Comparison of fitted values
myhat
yhat
y
x