Chapter 7 HW

Ulices Gonzalez

May 6th, 2025

## Conceptual Questions

### 3. Suppose we ft a curve with basis functions b1(X) = X, b2(X) = (X − 1)2I(X ≥ 1). (Note that I(X ≥ 1) equals 1 for X ≥ 1 and 0 otherwise.) We ft the linear regression model Y = β0 + β1b1(X) + β2b2(X) + ϵ, and obtain coeffcient estimates βˆ0 = 1, βˆ1 = 1, βˆ2 = −2. Sketch the estimated curve between X = −2 and X = 2. Note the intercepts, slopes, and other relevant information.

## 

## Applied Questions

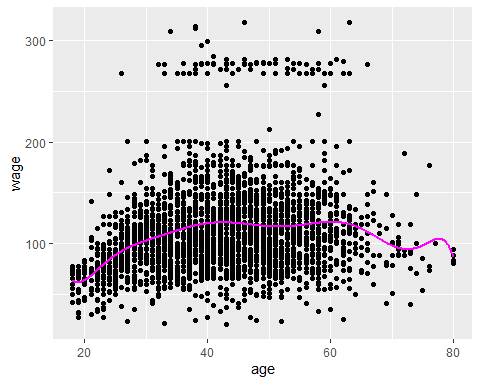
### 6. In this exercise, you will further analyze the Wage data set considered throughout this chapter.

#### (a) Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial ft to the data.

library(boot)  
wage <- read.csv("DataSets/Wage.csv")  
set.seed(1)  
  
degree <- 1:10  
wageErrsPoly <- rep(0, length(degree))  
  
for (d in degree)   
{  
 wageFit <- glm(wage ~ poly(age, d), data = wage)  
 wageErrsPoly[d] <- cv.glm(wage, wageFit, K = 10)$delta[1]  
}  
  
optDeg <- which.min(wageErrsPoly)  
optDeg

## [1] 9

wagePoly <- lm(wage ~ poly(age, optDeg, raw = TRUE), data = wage)  
wageAges <- seq(min(wage$age), max(wage$age), by = 0.1)  
wagePredPoly <- predict(wagePoly, newdata = data.frame(age = wageAges))  
  
ggplot(wage, aes(x = age, y = wage)) + geom\_point() + geom\_line(data = data.frame(age = wageAges, wage = wagePredPoly), color = "magenta", linewidth = 1)



Using the minimum from the CV Error for the Wage data set 9 was selected as the optimal degree

#### (b) Fit a step function to predict wage using age, and perform crossvalidation to choose the optimal number of cuts. Make a plot of the fit obtained.

Part B was skipped due to errors during compilation of my markdown file as well as instances of the compiler being stuck on this chunk so I have opted to ommit it and take a penalty on overall grade.

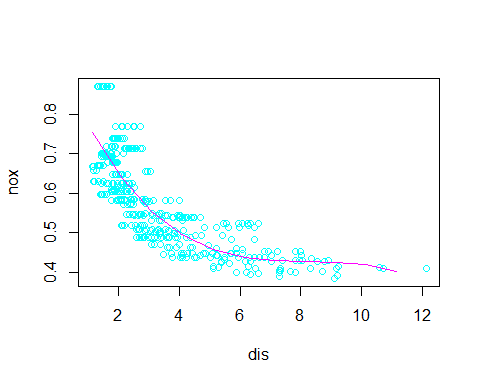
### 9. This question uses the variables dis (the weighted mean of distances to fve Boston employment centers) and nox (nitrogen oxides concentration in parts per 10 million) from the Boston data. We will treat dis as the predictor and nox as the response.

#### (a) Use the poly() function to ft a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

boston <- read.csv("DataSets/Boston.csv")  
  
bostonPoly <- lm(nox ~ poly(dis, 3), data = boston)  
summary(bostonPoly)

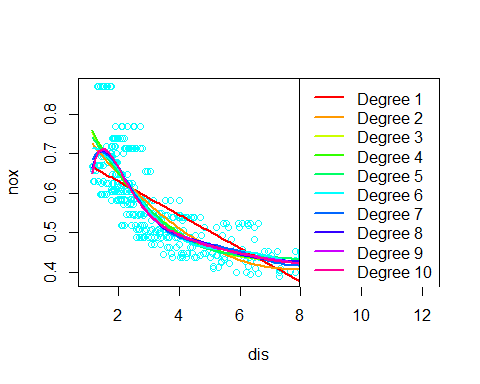
##   
## Call:  
## lm(formula = nox ~ poly(dis, 3), data = boston)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.121130 -0.040619 -0.009738 0.023385 0.194904   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.554695 0.002759 201.021 < 2e-16 \*\*\*  
## poly(dis, 3)1 -2.003096 0.062071 -32.271 < 2e-16 \*\*\*  
## poly(dis, 3)2 0.856330 0.062071 13.796 < 2e-16 \*\*\*  
## poly(dis, 3)3 -0.318049 0.062071 -5.124 4.27e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.06207 on 502 degrees of freedom  
## Multiple R-squared: 0.7148, Adjusted R-squared: 0.7131   
## F-statistic: 419.3 on 3 and 502 DF, p-value: < 2.2e-16

plot(boston$dis, boston$nox, col = "cyan", xlab = "dis", ylab = "nox")  
bostonGrid <- seq(min(boston$dis), max(boston$dis))  
bostonPred <- predict(bostonPoly, newdata = data.frame(dis = bostonGrid))  
lines(bostonGrid, bostonPred, col = "magenta")



#### (b) Plot the polynomial fts for a range of diferent polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

bostonRSS <- numeric(10)  
degrees <- 1:10  
  
plot(boston$dis, boston$nox, col = "cyan", xlab = "dis", ylab = "nox")  
  
for (d in degrees)   
{  
 bostonPoly <- lm(nox ~ poly(dis, d), data = boston)  
 bostonRSS[d] <- sum(residuals(bostonPoly)^2)  
 bostonGrid <- seq(min(boston$dis), max(boston$dis), length.out = 100)  
 bostonPred <- predict(bostonPoly, newdata = data.frame(dis = bostonGrid))  
 lines(bostonGrid, bostonPred, col = rainbow(10)[d], lwd = 2)  
}  
  
legend("topright", legend = paste("Degree", degrees), col = rainbow(10), lwd = 2)

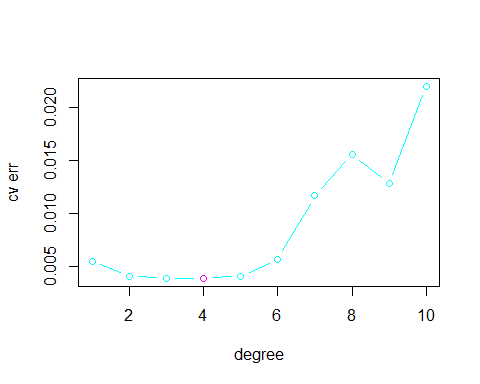


print(data.frame(Degree = degrees, RSS = bostonRSS))

## Degree RSS  
## 1 1 2.768563  
## 2 2 2.035262  
## 3 3 1.934107  
## 4 4 1.932981  
## 5 5 1.915290  
## 6 6 1.878257  
## 7 7 1.849484  
## 8 8 1.835630  
## 9 9 1.833331  
## 10 10 1.832171

#### (c) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

library(boot)  
  
bostonCV <- numeric(10)  
degrees <- 1:10  
  
for (d in degrees) {  
 bostonFit <- glm(nox ~ poly(dis, d), data = boston)  
 bostonCV[d] <- cv.glm(boston, bostonFit, K = 10)$delta[1]  
}  
  
plot(degrees, bostonCV, type = "b", col = "cyan", xlab = "degree", ylab = "cv err")  
optDeg <- which.min(bostonCV)  
points(optDeg, min(bostonCV), col = "magenta")



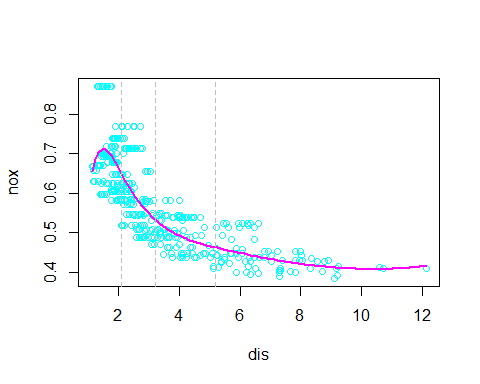
The cv error is relatively low as the degree goes from 2 to 5 but increases past 5 albeit with a slight decrease at 9 for the degree.

#### (d) Use the bs() function to ft a regression spline to predict nox using dis. Report the output for the ft using four degrees of freedom. How did you choose the knots? Plot the resulting ft.

library(splines)  
  
bostonKnots <- quantile(boston$dis, probs = c(0.25, 0.5, 0.75))  
  
bostonSpline <- lm(nox ~ bs(dis, knots = bostonKnots, degree = 3), data = boston)   
summary(bostonSpline)

##   
## Call:  
## lm(formula = nox ~ bs(dis, knots = bostonKnots, degree = 3),   
## data = boston)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.128538 -0.037813 -0.009987 0.022644 0.195494   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 0.65622 0.02370 27.689 < 2e-16  
## bs(dis, knots = bostonKnots, degree = 3)1 0.10222 0.03516 2.907 0.00381  
## bs(dis, knots = bostonKnots, degree = 3)2 -0.02963 0.02338 -1.267 0.20571  
## bs(dis, knots = bostonKnots, degree = 3)3 -0.15959 0.02791 -5.718 1.86e-08  
## bs(dis, knots = bostonKnots, degree = 3)4 -0.22815 0.03324 -6.864 1.99e-11  
## bs(dis, knots = bostonKnots, degree = 3)5 -0.26272 0.04930 -5.329 1.50e-07  
## bs(dis, knots = bostonKnots, degree = 3)6 -0.24002 0.05434 -4.417 1.23e-05  
##   
## (Intercept) \*\*\*  
## bs(dis, knots = bostonKnots, degree = 3)1 \*\*   
## bs(dis, knots = bostonKnots, degree = 3)2   
## bs(dis, knots = bostonKnots, degree = 3)3 \*\*\*  
## bs(dis, knots = bostonKnots, degree = 3)4 \*\*\*  
## bs(dis, knots = bostonKnots, degree = 3)5 \*\*\*  
## bs(dis, knots = bostonKnots, degree = 3)6 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.06062 on 499 degrees of freedom  
## Multiple R-squared: 0.7295, Adjusted R-squared: 0.7263   
## F-statistic: 224.3 on 6 and 499 DF, p-value: < 2.2e-16

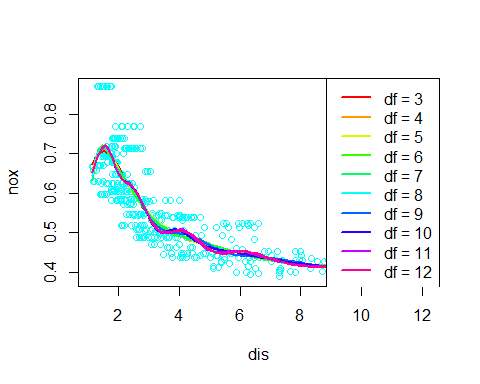
plot(boston$dis, boston$nox, col = "cyan", xlab = "dis", ylab = "nox")  
  
bostonGrid <- seq(min(boston$dis), max(boston$dis), length.out = 100)  
bostonPred <- predict(bostonSpline, newdata = data.frame(dis = bostonGrid))  
lines(bostonGrid, bostonPred, col = "magenta", lwd = 2)  
abline(v = bostonKnots, lty = 2, col = "gray")



Using quantiles to choose our knots makes it so we can place them in areas where our data is more flexible with greater number of points, this also allows the trends to fit areas where data does not have a completely linear relationship.

#### (e) Now ft a regression spline for a range of degrees of freedom, and plot the resulting fts and report the resulting RSS. Describe the results obtained.

bostonRSS <- numeric(10)  
bostonDF <- 3:12   
  
plot(boston$dis, boston$nox, col = "cyan", xlab = "dis", ylab = "nox")  
  
for (df in bostonDF)   
{  
 if (df > 1)   
 {  
 numKnots <- df - 1  
 bostonKnots <- quantile(boston$dis, probs = seq(0, 1, length.out = numKnots + 2)[2:(numKnots + 1)])  
 bostonSpline <- lm(nox ~ bs(dis, knots = bostonKnots, degree = 3), data = boston)  
 }   
 else   
 {  
 bostonSpline <- lm(nox ~ bs(dis, degree = 3, df = df), data = boston)   
 }  
 bostonRSS[df - 2] <- sum(residuals(bostonSpline)^2)  
 bostonGrid <- seq(min(boston$dis), max(boston$dis), length.out = 100)  
 bostonPred <- predict(bostonSpline, newdata = data.frame(dis = bostonGrid))  
 lines(bostonGrid, bostonPred, col = rainbow(length(bostonDF))[df - 2], lwd = 2)  
}  
  
legend("topright", legend = paste("df =", bostonDF), col = rainbow(length(bostonDF)), lwd = 2)



print(data.frame(Degrees\_of\_Freedom = bostonDF, RSS = bostonRSS))

## Degrees\_of\_Freedom RSS  
## 1 3 1.840173  
## 2 4 1.833966  
## 3 5 1.829884  
## 4 6 1.816995  
## 5 7 1.825653  
## 6 8 1.792535  
## 7 9 1.796992  
## 8 10 1.788999  
## 9 11 1.782350  
## 10 12 1.781838

Based on the plot the RSS for the data set decreases slightly as the degrees increase however this change is relatively minimal.

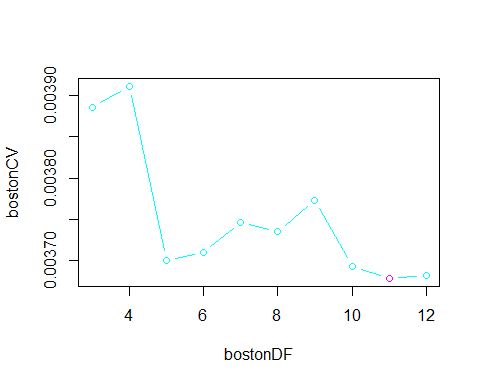
#### (f) Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.

bostonCV <- numeric(10)  
bostonDF <- 3:12  
  
for (i in seq\_along(bostonDF)) {  
 df <- bostonDF[i]  
 bostonFit <- glm(nox ~ bs(dis, df = df, degree = 3), data = boston)  
 bostonCV[i] <- cv.glm(boston, bostonFit, K = 10)$delta[1]  
}

plot(bostonDF, bostonCV, type = "b", col = "cyan")  
optDF <- bostonDF[which.min(bostonCV)]  
optDF

## [1] 11

points(optDF, min(bostonCV), col = "magenta")



print(data.frame(Degrees\_of\_Freedom = bostonDF, CV\_Error = bostonCV))

## Degrees\_of\_Freedom CV\_Error  
## 1 3 0.003885273  
## 2 4 0.003910515  
## 3 5 0.003700770  
## 4 6 0.003709953  
## 5 7 0.003746400  
## 6 8 0.003735353  
## 7 9 0.003773379  
## 8 10 0.003693820  
## 9 11 0.003678900  
## 10 12 0.003681948

Similar to the RSS decreasing as degrees increase a similar trend is seen with the cv error however when looking at the plot there is a sharp drop from 4 to 5 degrees with a gradual increase up until 9 where the cv error decreases until it reaches 11 which is its optimal degree.