STATS415hw8

Yunguo Cai

3/16/2018

(a)

```
set.seed(34567)
train = sample(1:nrow(Boston),floor(nrow(Boston)*0.8))
Boston_train = Boston[train, ]
Boston_test = Boston[-train, ]
```

(b)

```
#polynomial regression
set.seed(34567)
cv.error_poly = rep(0,17)
for (i in 1:17){
  fit_poly=glm(nox~poly(dis,i),data=Boston_train)
  cv.error_poly[i]=cv.glm(Boston_train, fit_poly, K=10)$delta[1]
}
cv.error_poly
```

```
##  [1] 0.005647038 0.004172661 0.003950147 0.003976680 0.004014418
##  [6] 0.003803607 0.003678527 0.003685583 0.003687592 0.003778827
## [11] 0.008162215 0.003760644 0.003728638 0.003705171 0.003647734
## [16] 0.003615007 0.003635486
```

```
fit.1=lm(nox~dis,data=Boston_train)
fit.2=lm(nox~poly(dis,2),data=Boston_train)
fit.3=lm(nox~poly(dis,3),data=Boston_train)
fit.4=lm(nox~poly(dis,4),data=Boston_train)
fit.7=lm(nox~poly(dis,7),data=Boston_train)
anova(fit.1,fit.2,fit.3,fit.4,fit.7)
```

```
## Analysis of Variance Table
##
## Model 1: nox ~ dis
## Model 2: nox ~ poly(dis, 2)
## Model 3: nox ~ poly(dis, 3)
## Model 4: nox ~ poly(dis, 4)
## Model 5: nox ~ poly(dis, 7)
##   Res.Df    RSS Df Sum of Sq        F    Pr(>F)
## 1    402 2.2537
## 2    401 1.6601  1   0.59359 160.4977 < 2.2e-16 ***
## 3    400 1.5732  1   0.08687  23.4874 1.808e-06 ***
## 4    399 1.5719  1   0.00128   0.3455     0.557
## 5    396 1.4646  3   0.10735   9.6753 3.551e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fit.poly=lm(nox~poly(dis,7),data=Boston_train)
summary(fit.poly)
```

```
##
## Call:
```

```
## lm(formula = nox ~ poly(dis, 7), data = Boston_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13328 -0.03883 -0.01079  0.02668  0.20298
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.553415   0.003026 182.907  < 2e-16 ***
## poly(dis, 7)1 -1.770681   0.060815 -29.116  < 2e-16 ***
## poly(dis, 7)2  0.770450   0.060815  12.669  < 2e-16 ***
## poly(dis, 7)3 -0.294733   0.060815  -4.846 1.81e-06 ***
## poly(dis, 7)4  0.035747   0.060815   0.588 0.557000
## poly(dis, 7)5  0.140129   0.060815   2.304 0.021729 *
## poly(dis, 7)6 -0.212277   0.060815  -3.491 0.000536 ***
## poly(dis, 7)7  0.206528   0.060815   3.396 0.000753 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06081 on 396 degrees of freedom
## Multiple R-squared:  0.7282, Adjusted R-squared:  0.7234
## F-statistic: 151.6 on 7 and 396 DF,  p-value: < 2.2e-16
```

For the polynomial regression, choose the degree of freedom of 7. Though degree=16 gives the least error, from the hypothesis test we can see that the cv error doesn't vary much after 7 and p value is significant at the confidence level of 0.05, so I choose degree=7.

```
#natural spline
set.seed(34567)
cv.error_ns = rep(0,15)
for (i in 1:15){
  fit_ns=glm(nox~ns(dis,df = i),data=Boston_train)
  cv.error_ns[i]=cv.glm(Boston_train, fit_ns, K=10)$delta[1]
}
which.min(cv.error_ns)
```

```
## [1] 9
```

```
cv.error_ns
```

```
##  [1] 0.005647038 0.004031811 0.003930074 0.003911924 0.003840186
##  [6] 0.003782349 0.003740438 0.003615852 0.003603003 0.003707900
## [11] 0.003673059 0.003690553 0.003729949 0.003709152 0.003727432
```

For the natural spline, choose the degree of freedom of 9. The cv error goes down first and then increases, which indicates that the model has bigger bias at first and then fits better and after dof=9, it becomes overfitting.

```
#smoothing spline
set.seed(34567)
fit_ss=smooth.spline(Boston_train$dis,Boston_train$nox,cv=TRUE)
```

```
## Warning in smooth.spline(Boston_train$dis, Boston_train$nox, cv = TRUE):
## cross-validation with non-unique 'x' values seems doubtful
```

```
fit_ss
```

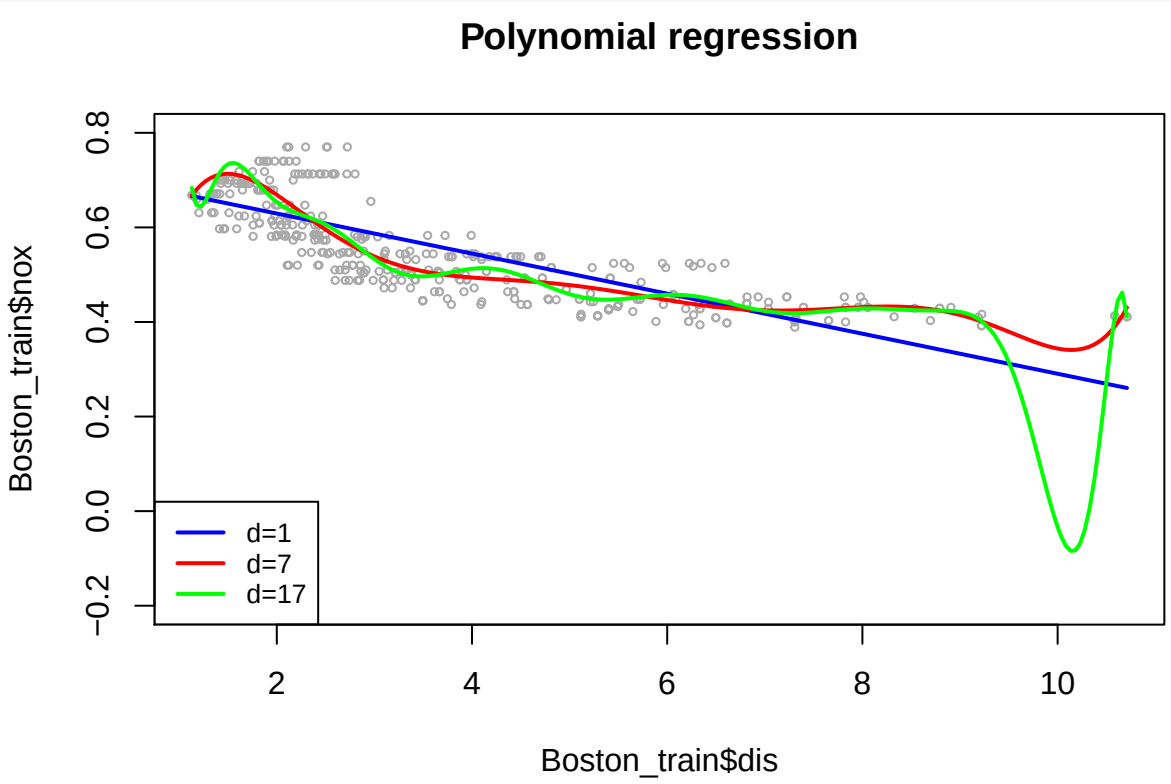
```
## Call:
```

```
## smooth.spline(x = Boston_train$dis, y = Boston_train$nox, cv = TRUE)
##
## Smoothing Parameter  spar= 0.8273621  lambda= 7.566141e-05 (11 iterations)
## Equivalent Degrees of Freedom (Df): 15.98821
## Penalized Criterion (RSS): 1.396404
## PRESS(l.o.o. CV): 0.003646622
```

For the smooth spline, choose the degree of freedom of 15.99.$\lambda = 7.57 \times 10^{-5}$. The error is 0.00365.

(c)

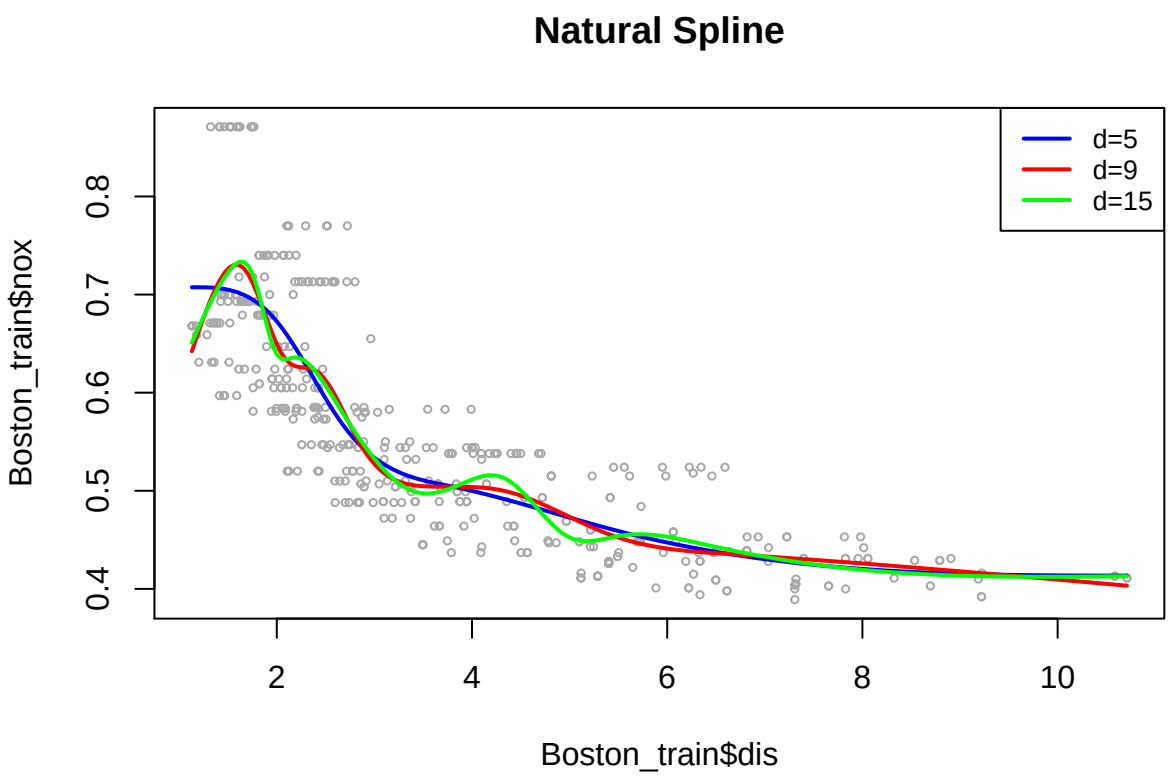
```
#polynomial regression
dislims=range(Boston_train$dis)
dis.grid=seq(from=dislims[1],to=dislims[2],length.out = 200)
fit.17=lm(nox~poly(dis,17),data=Boston_train)
preds1=predict(fit.1,newdata=data.frame(dis=dis.grid))
preds7=predict(fit.poly,newdata=data.frame(dis=dis.grid))
preds17=predict(fit.17,newdata=data.frame(dis=dis.grid))
plot(Boston_train$dis,Boston_train$nox,xlim=dislims,ylim=-0.2:1,cex=.5,col="darkgrey")
title("Polynomial regression")
lines(dis.grid,preds1,lwd=2,col="blue")
lines(dis.grid,preds7,lwd=2,col="red")
lines(dis.grid,preds17,lwd=2,col="green")
legend("bottomleft",legend=c("d=1","d=7","d=17"),col=c("blue","red","green"),lty=1,lwd=2,cex=.8)
```



In the polynomial regression, when d is smaller than what we choose (when d=1), the fit is smooth and linear and the bias is bigger. As d increases, the model fits the training points better, but the variance also gets bigger. So with the further increase of d, the model becomes overfitting (like the one when d=17).

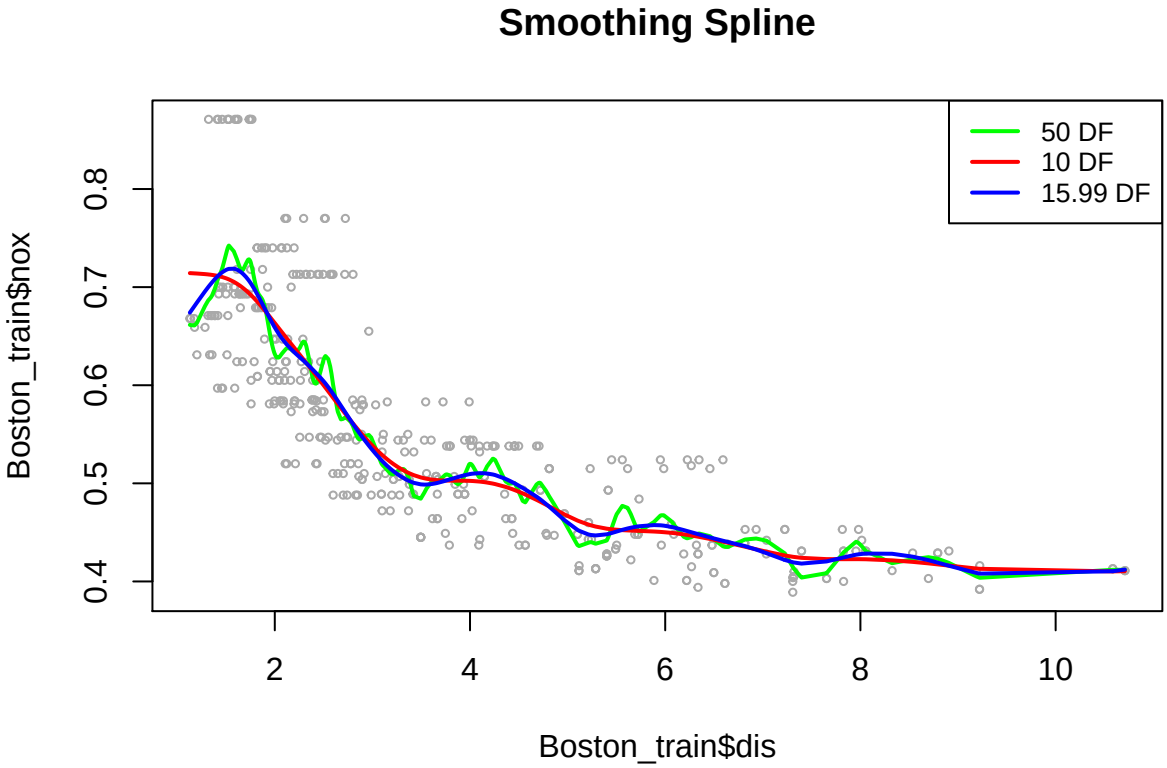
```
#natural spline
fit.5=glm(nox~ns(dis,df = 5),data=Boston_train)
fit_ns=glm(nox~ns(dis,df = 9),data=Boston_train)
```

```
fit.15=glm(nox~ns(dis,df = 15),data=Boston_train)
preds_5=predict(fit.5,newdata=data.frame(dis=dis.grid))
preds_9=predict(fit_ns,newdata=data.frame(dis=dis.grid))
preds_15=predict(fit.15,newdata=data.frame(dis=dis.grid))
plot(Boston_train$dis,Boston_train$nox,xlim=dislims,cex=.5,col="darkgrey")
title("Natural Spline")
lines(dis.grid,preds_5,lwd=2,col="blue")
lines(dis.grid,preds_9,lwd=2,col="red")
lines(dis.grid,preds_15,lwd=2,col="green")
legend("topright",legend=c("d=5","d=9","d=15"),col=c("blue","red","green"),lty=1,lwd=2,cex=.8)
```



For the nature spline, when d is smaller than what we choose, the regression is smoother and has bigger bias. With the increase of degree of freedom, the model fits better and gets more sensitive to fluctuation. When dof becomes too big (like dof=15), it becomes overfitting.

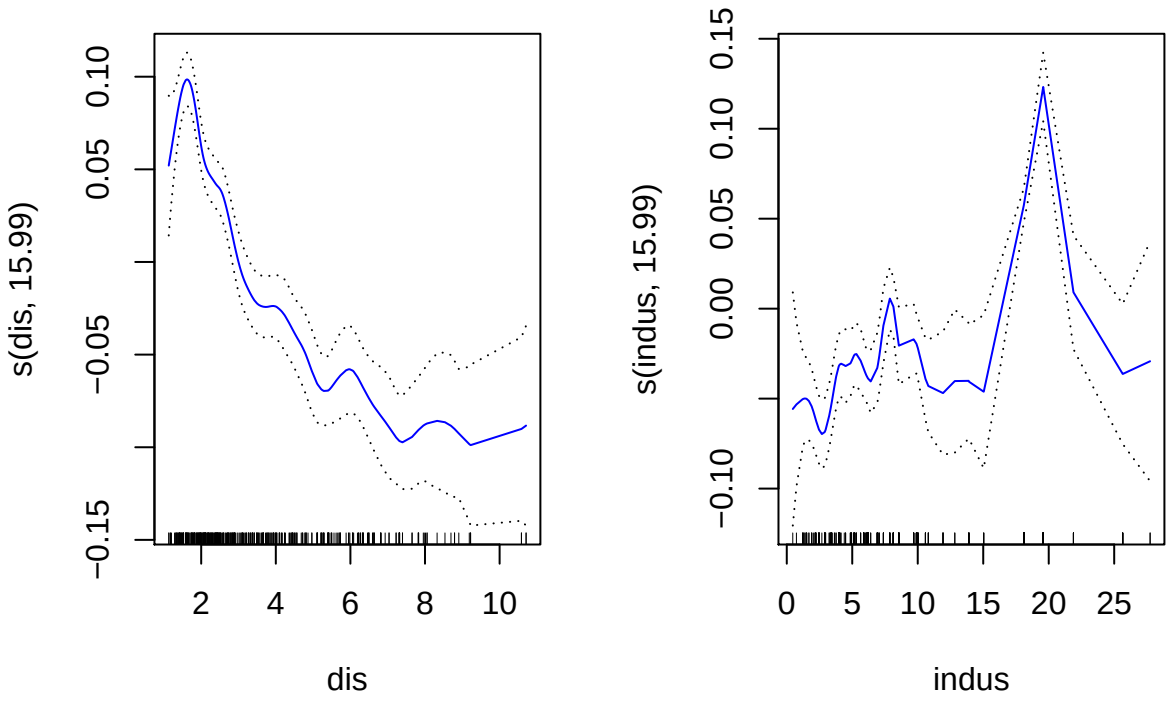
```
#smoothing spline
fit.10=smooth.spline(Boston_train$dis,Boston_train$nox,df=10)
fit.50=smooth.spline(Boston_train$dis,Boston_train$nox,df=50)
plot(Boston_train$dis,Boston_train$nox,xlim=dislims,cex=.5,col="darkgrey")
title("Smoothing Spline")
lines(fit.50,col="green",lwd=2)
lines(fit.10,col="red",lwd=2)
lines(fit_ss,col="blue",lwd=2)
legend("topright",legend=c("50 DF","10 DF","15.99 DF"),col=c("green","red","blue"),lty=1,lwd=2,cex=.8)
```



For the smooth spline, it performs similar to nature spline. But it fits better with respect to different area between knots. With the increase of d, the model first fits better and then get overfitting. In all, smooth spline performs better than the other two from cv.

(d)

```
set.seed(34567)
gam=gam(nox~s(dis,15.99)+s(indus,15.99),data=Boston_train)
par(mfrow=c(1,2))
plot(gam, se=TRUE,col="blue")
```



(e)

```
test.poly = predict(fit.poly,Boston_test)
test.ns = predict(fit_ns,Boston_test)
test.ss = predict(fit_ss,dis[-train])
test.gam = predict(gam,Boston_test)
nox.test=nox[-train]
error.poly = mean((nox.test-test.poly)^2)
error.ns = mean((nox.test-test.ns)^2)
error.ss = mean((nox.test-test.ss$y)^2)
error.gam = mean((nox.test-test.gam)^2)
d <- data.frame("TestMSE" = c(error.poly, error.ns, error.ss, error.gam))
rownames(d) <- c("poly regression","natural spline", "smoothing spline"," GAM")
knitr::kable(d)
```

	TestMSE
poly regression	0.0908248
natural spline	0.0040118
smoothing spline	0.0038431
GAM	0.0024323

From the table, it shows that GAM method gives the least error, thus the best performance in fitting the model. Natural and smoothing spline give similar results, so we can conclude that the ends of data points don't vary much. Polynomial regression has the highest test error, which may be caused by the boundary behaviors, because from the plot in (c) it shows that around dis = 10, the curve has an immediate increase.