

Praktikum 9

A1

a)

A2

a)

Sei λ fest.

$$\hat{f}(x) = \left(\sum_{i=1}^n K_\lambda(x, x_i) y_i \right) \left(\sum_{i=1}^n K_\lambda(x, x_i) \right)^{-1}$$

Der Tricube-Kern ist definiert als

$$K_\lambda(x, x_i) = \begin{cases} \left(1 - \left|\frac{x-x_i}{\lambda}\right|^3\right)^3, & |x - x_i| \leq \lambda \\ 0, & \text{sonst.} \end{cases}$$

Für x_i fest ist $K_\lambda(x, x_i)$ für jedes x_i ohne $|x - x_i| = \lambda$ als Verknüpfung differenzierbarer Funktionen differenzierbar. Folglich bleiben die Punkte

$$|x - x_i| = \lambda \implies x_0 = \lambda + x_i \text{ und } x_0 = x_i - \lambda$$

als kritische Stellen auf Stetigkeit zu prüfen. Der Kern ist offensichtlich stetig in x_0 , da der Wert beidseitig problemlos gegen 0 konvergiert. Wir prüfen die Differenzierbarkeit:

Sei $x_0 = \lambda + x_i$.

$$\lim_{h \rightarrow 0} = \frac{1}{h} \left(\left(1 - \left|\frac{\lambda + x_i - x_i + h}{\lambda}\right|^3\right)^3 - \left(1 - \left|\frac{\lambda + x_i - x_i}{\lambda}\right|^3\right) \right) \quad (1)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(\left(1 - \left|1 + \frac{h}{\lambda}\right|^3\right)^3 - (1 - 1)^3 \right) \quad (2)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(1 - \left(1 + \frac{h}{\lambda}\right)^3 \right)^3 \quad (3)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(1 - \left(1 + 3\frac{h}{\lambda} + 3\frac{h^2}{\lambda^2} + \frac{h^3}{\lambda^3}\right)^3 \right) \quad (4)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(3\frac{h}{\lambda} + 3\frac{h^2}{\lambda^2} + \frac{h^3}{\lambda^3} \right)^3 = 0 \quad (5)$$

und der linksseitige Beweis ($-h$) folgt analog.

Sei $x_0 = x_i - \lambda$.

$$\lim_{h \rightarrow 0} = \frac{1}{h} \left(\left(1 - \left| \frac{x_i - \lambda - x_i + h}{\lambda} \right|^3 \right)^3 - \left(1 - \left| \frac{-\lambda + x_i - x_i}{\lambda} \right|^3 \right) \right) \quad (6)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(\left(1 - \underbrace{\left| -1 + \frac{h}{\lambda} \right|}_{<0 \text{ für } h \rightarrow 0} \right)^3 - (1 - 1)^3 \right) \quad (7)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(1 - \left(1 - \frac{h}{\lambda} \right)^3 \right) \quad (8)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(1 - \left(1 - 3\frac{h}{\lambda} + 3\frac{h^2}{\lambda^2} - \frac{h^3}{\lambda^3} \right)^3 \right) \quad (9)$$

$$= \lim_{h \rightarrow 0} \frac{1}{h} \left(-3\frac{h}{\lambda} + 3\frac{h^2}{\lambda^2} - \frac{h^3}{\lambda^3} \right)^3 = 0 \quad (10)$$

und der linksseitige Beweis $(-h)$ folgt analog.

Somit ist $K_\lambda(x, x_i)$ stetig differenzierbar und folglich ist $\hat{f}(x)$ ist Verkettung differenzierbarer Funktionen differenzierbar.

b)

Nimmt man nun den Epanechnikov Kernel

$$K_\lambda(x, x_i) = \begin{cases} \frac{3}{4} \left(1 - \left(\frac{x - x_i}{\lambda} \right)^2 \right), & |x - x_i| \leq \lambda \\ 0, & \text{sonst.} \end{cases}$$

dann sieht man, dass der Beweis der differenzierbarkeit analog verläuft. Der einzige nennenswerte Unterschied ist der Wegfall der beiden kubischen Potenzen zu einer quadratischen Potenz.

Somit bleibt in Schritt (10) ein Term der Form

$$\lim_{h \rightarrow 0} \frac{1}{h} a \left(b\frac{h}{\lambda} + c\frac{h^2}{\lambda^2} \right), \quad a, b, c \in \mathbb{R}$$

und dieser konvergiert wieder problemlos gegen 0 und wir erhalten die Differenzierbarkeit.

c)

Ersetzen wir im Beweis der Differenzierbarkeit alle Vorkommen von λ durch eine variable Funktion $h_\lambda(x) > 0$, so verändert sich im Beweis nichts grundlegend. Die Limiten konvergieren und der Kernel Smoother bleibt differenzierbar.

A3)

```
set.seed(1337)
```

a)

```
data(ethanol)

loessMod10 <- loess(NOx ~ E, data=ethanol, span=0.1) # 10% smoothing span
loessMod30 <- loess(NOx ~ E, data=ethanol, span=0.3) # 25% smoothing span
loessMod50 <- loess(NOx ~ E, data=ethanol, span=0.5)

spans = seq(0.1, 1, 0.02)

#Randomly shuffle the data
data <- ethanol[sample(nrow(ethanol)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(data)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
test <- matrix(NA, ncol=2, nrow = length(spans))
mse1 <- c()
for (span in spans){
  mse <- c()
  for(j in 1:5){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==j,arr.ind=TRUE)
    testData <- data[testIndexes, ]
    trainData <- data[-testIndexes, ]
    #Use the test and train data partitions however you desire...

    fit <- loess(NOx ~ E, data=trainData, span=span, control = loess.control(surface="direct"))

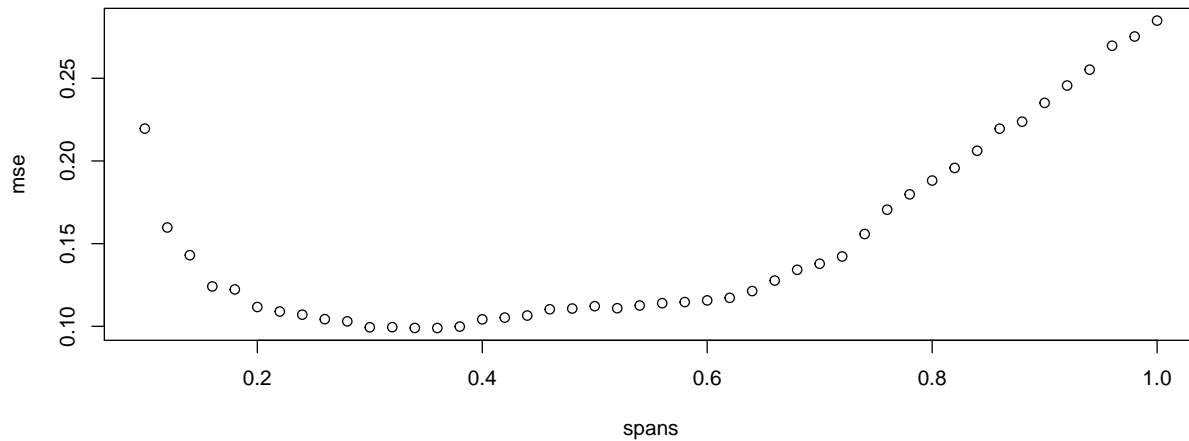
    y.pred <- predict(fit, newdata = testData$E)
    y.true = testData$NOx

    mse <- append(mse, mean((y.pred-y.true)^2))

  }
  mse1 <- append(mse1, mean(mse))
}

test <- NULL
test$spans <- spans
test$mse <- mse1
test <- as.data.frame(test)

plot(test)
```

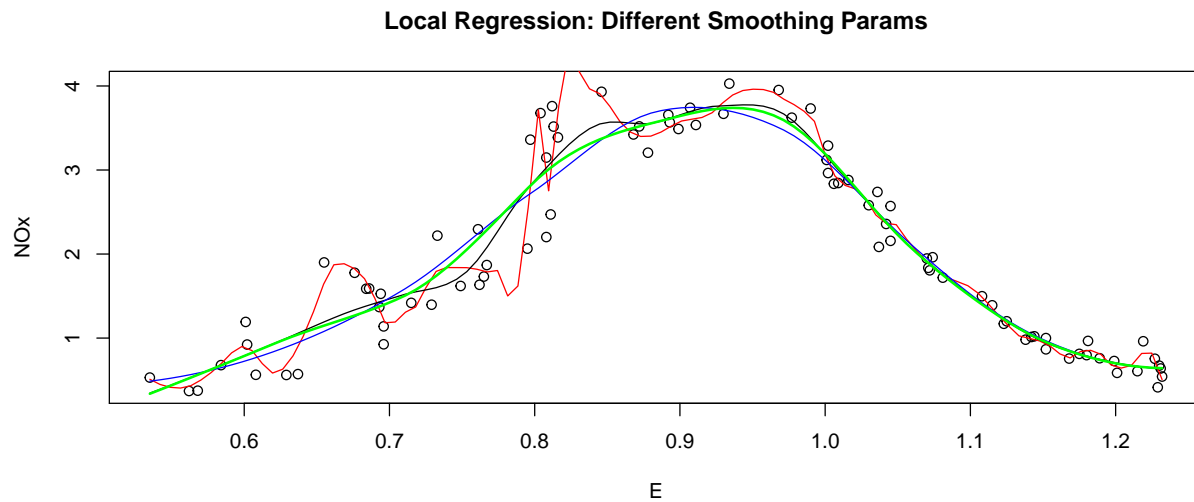


```
opt_spans <- test$spans[test$mse == min(test[, "mse"])]
opt_spans
#> [1] 0.36
loessMod_opt <- loess(NOx ~ E, data=ethanol, span=opt_spans)
```

b)

```
tricube_10 <- locfit(NOx~lp(E,nn=0.1), data=ethanol)
tricube_30 <- locfit(NOx~lp(E,nn=0.3), data=ethanol)
tricube_50 <- locfit(NOx~lp(E,nn=0.5), data=ethanol)
tricube_opt <- locfit(NOx~lp(E,nn=opt_spans), data=ethanol)

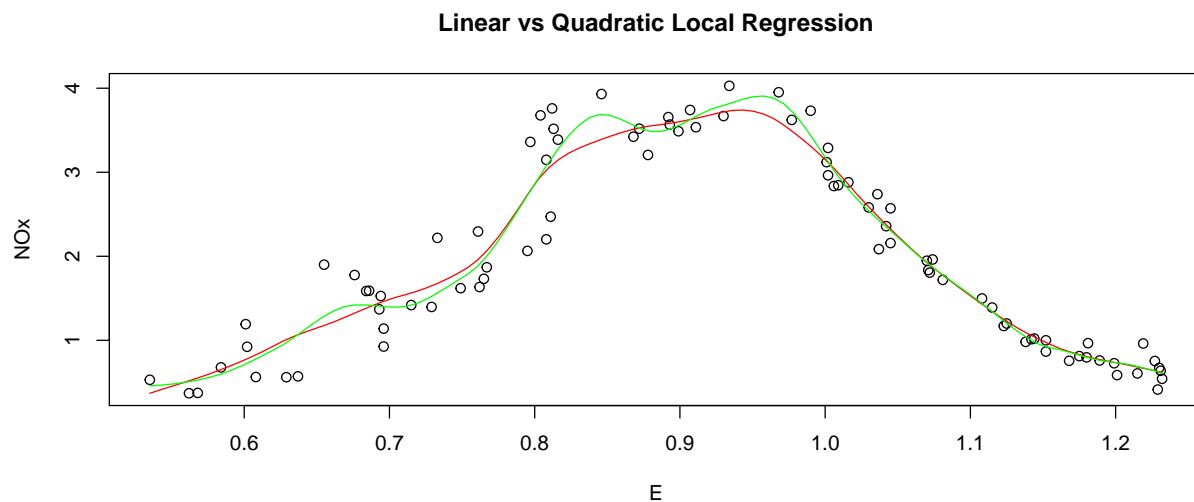
plot(ethanol$NOx, x=ethanol$E, main="Local Regression: Different Smoothing Params", xlab="E", ylab="NOx")
lines(tricube_10, col="red")
lines(tricube_30, col="black")
lines(tricube_50, col="blue")
lines(tricube_opt, col="green", lw=2)
```



c)

```
lin <- locfit(NOx ~ E, data=ethanol, alpha = 0.2, deg=1)
quad <- locfit(NOx ~ E, data=ethanol, alpha = 0.2, deg=2)

plot(ethanol$NOx, x=ethanol$E, main="Linear vs Quadratic Local Regression", xlab="E", ylab="NOx")
lines(lin, col="red")
lines(quad, col="green")
```



```
lin_pred <- cbind(predict(lin, newdata = c(0.65)), predict(lin, newdata = c(0.9)))
quad_pred <- cbind(predict(quad, newdata = c(0.65)), predict(quad, newdata = c(0.9)))

preds <- as.data.frame(rbind(lin_pred, quad_pred), row.names = c("lin", "quad"))
colnames(preds) <- c("0.65", "0.9")
```

```
preds
#>           0.65           9
#> lin  1.146328 3.602708
#> quad 1.229701 3.562719
```

d)

```
mse1 <- c()
for (span in spans){
  mse <- c()
  for(j in 1:5){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==j,arr.ind=TRUE)
    testData <- data[testIndexes, ]
    trainData <- data[-testIndexes, ]
    #Use the test and train data partitions however you desire...

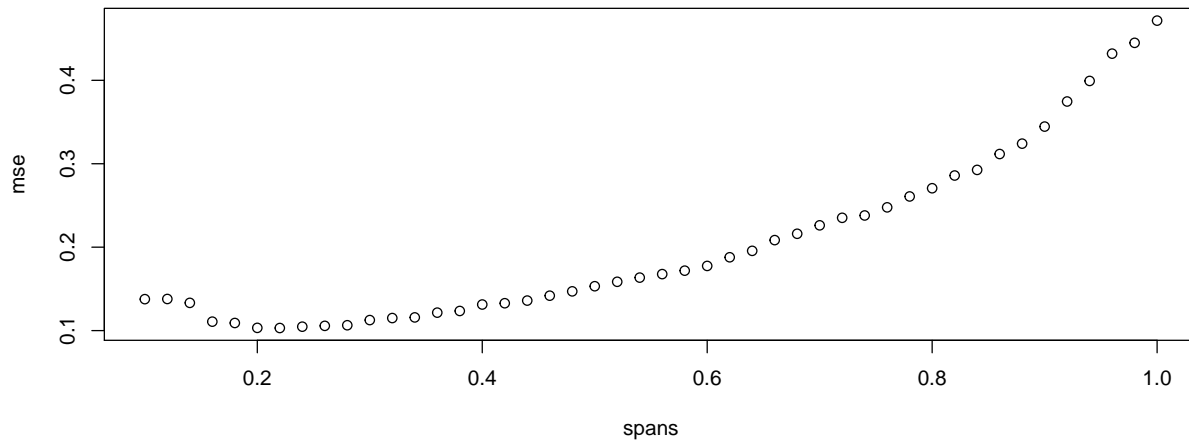
    fit <- locfit(NOx ~ E, data=trainData, alpha = span, deg=1)

    y.pred <- predict(fit, newdata = testData$E)
    y.true = testData$NOx

    mse <- append(mse, mean((y.pred-y.true)^2))
  }
  mse1 <- append(mse1, mean(mse))
}

lin_cv <- NULL
lin_cv$spans <- spans
lin_cv$mse <- mse1
lin_cv <- as.data.frame(lin_cv)

plot(lin_cv)
```



```

opt_lin <- lin_cv$spans[lin_cv$mse == min(lin_cv[, "mse"])]
opt_lin
#> [1] 0.22

mse1 <- c()
for (span in spans){
  mse <- c()
  for(j in 1:5){
    #Segment your data by fold using the which() function
    testIndexes <- which(folds==j,arr.ind=TRUE)
    testData <- data[testIndexes, ]
    trainData <- data[-testIndexes, ]
    #Use the test and train data partitions however you desire...

    fit <- locfit(NOx ~ E, data=trainData, alpha = span, deg=2)

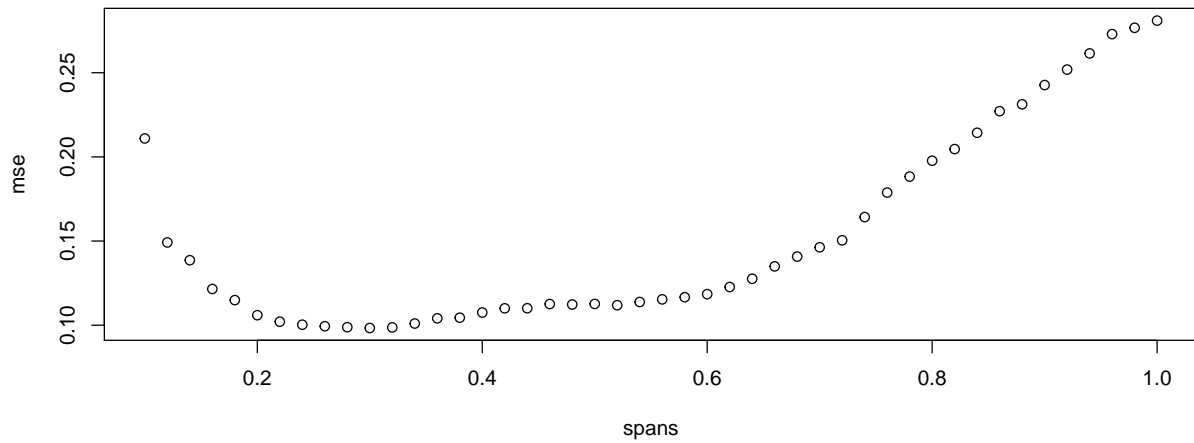
    y.pred <- predict(fit, newdata = testData$E)
    y.true = testData$NOx

    mse <- append(mse, mean((y.pred-y.true)^2))
  }
  mse1 <- append(mse1, mean(mse))
}

quad_cv <- NULL
quad_cv$spans <- spans
quad_cv$mse <- mse1
quad_cv <- as.data.frame(quad_cv)

plot(quad_cv)

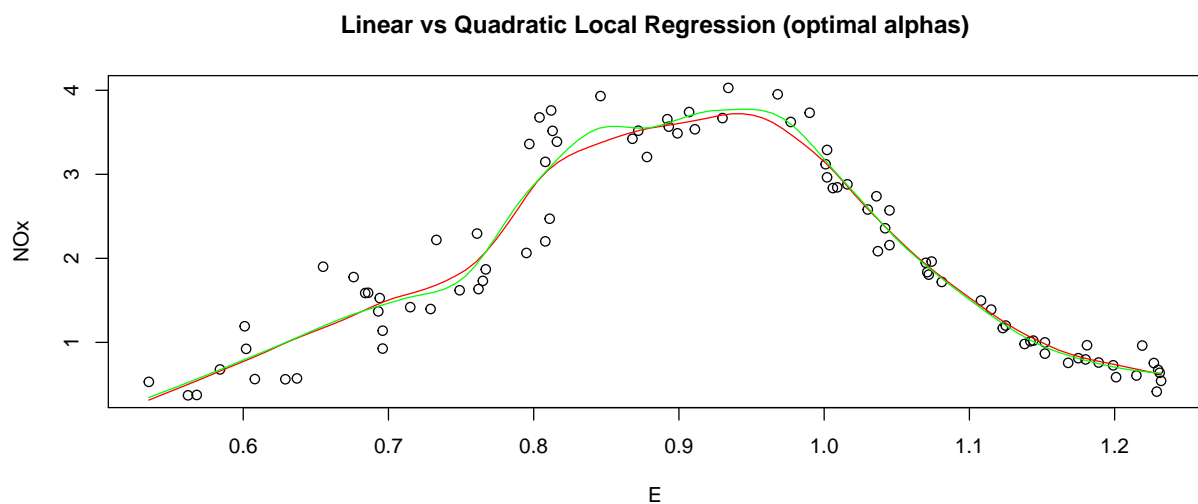
```

```
opt_quad <- quad_cv$spans[quad_cv$mse == min(quad_cv[, "mse"])]
opt_quad
#> [1] 0.3

lin_opt <- locfit(NOx ~ E, data=ethanol, alpha = opt_lin, deg=1)
quad_opt <- locfit(NOx ~ E, data=ethanol, alpha = opt_quad, deg=2)

plot(ethanol$NOx, x=ethanol$E, main="Linear vs Quadratic Local Regression (optimal alphas)", xlab="E", ylab="NOx",
     lines(lin_opt, col="red")
     lines(quad_opt, col="green"))
```



f)

```
sum(residuals(loessMod10)^2)
#> [1] 3.382258
sum(residuals(loessMod30)^2)
#> [1] 7.18373
sum(residuals(loessMod50)^2)
#> [1] 9.168455
sum(residuals(loessMod_opt)^2)
#> [1] 7.443603

sum(residuals(tricube_10)^2)
#> [1] 3.442665
sum(residuals(tricube_30)^2)
#> [1] 7.093455
sum(residuals(tricube_50)^2)
#> [1] 9.067521
sum(residuals(tricube_opt)^2)
#> [1] 7.976709

sum(residuals(lin_opt)^2)
#> [1] 7.481113
sum(residuals(quad_opt)^2)
#> [1] 7.093455
```